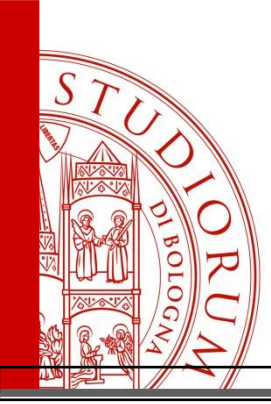


Università degli Studi di Bologna
Corso di Laurea in Ingegneria Informatica

Analisi del Problema

Ingegneria del Software T

Prof. MARCO PATELLA
Dipartimento di Informatica – Scienza e Ingegneria (DISI)

Sommario

- Introduzione
- Analisi del documento dei requisiti
- Analisi dei ruoli e delle responsabilità
- Scomposizione del problema
- Creazione modello del dominio
- Architettura Logica: Struttura
- Architettura Logica: Interazione
- Architettura Logica: Comportamento
- Definizione del Piano di Lavoro
- Definizione del Piano del Collaudo

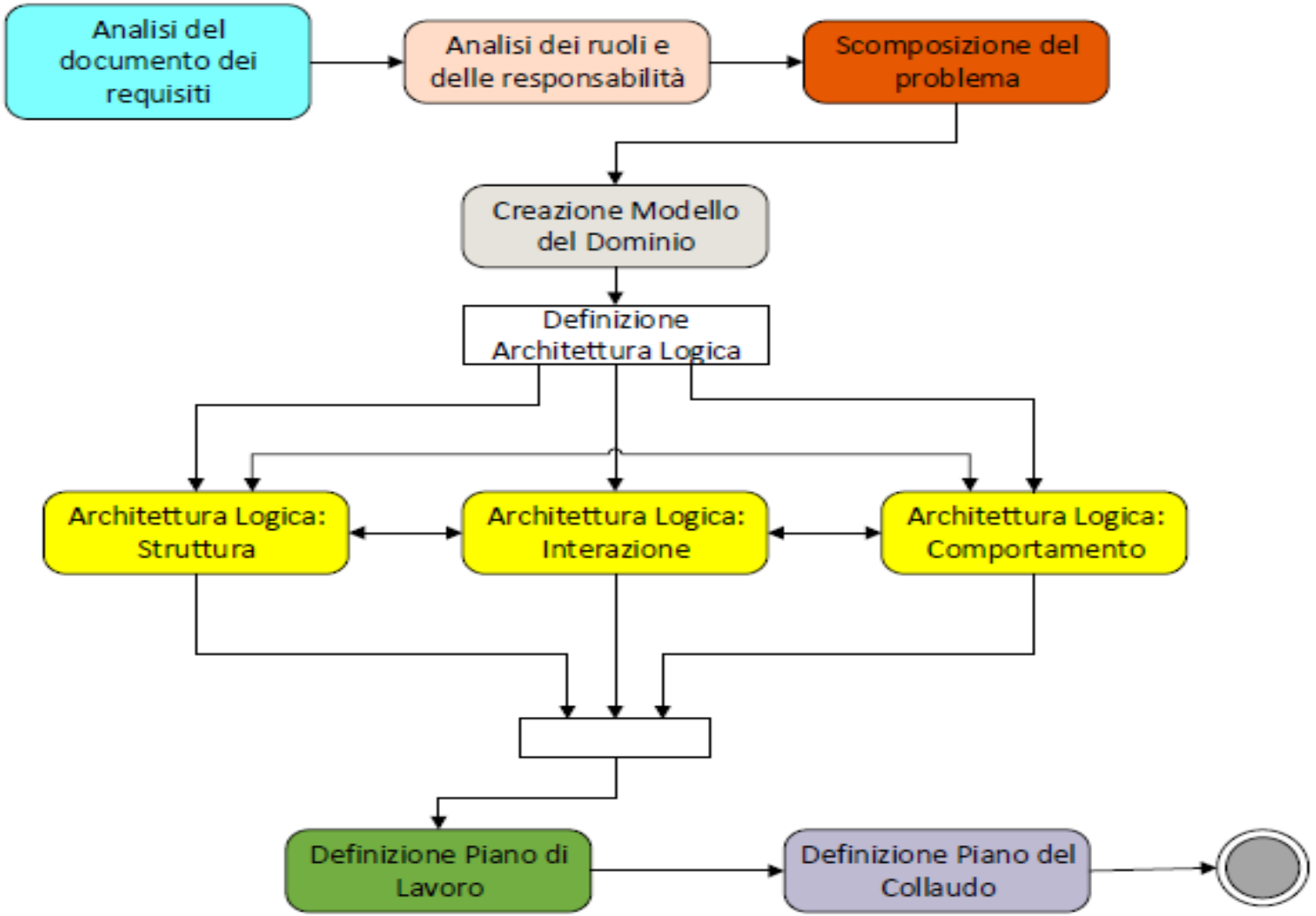
Introduzione

Analisi del Problema

- Obiettivo:
  esprimere fatti il più possibile “oggettivi”
  sul ***problema*** focalizzando l’attenzione su *sottosistemi*, *ruoli* e *responsabilità* insiti negli scenari
  prospettati durante l’analisi dei requisiti
  senza descrivere la sua possibile soluzione

- Risultato:
  - Architettura Logica
  - Piano di Lavoro
  - Piano del Collaudo

Analisi del Problema



Analisi del Problema

1. Analisi del Documento dei Requisiti
   Obiettivo: concentrarsi sull'analisi delle funzionalità e dei rischi evidenziati nel documento dei requisiti
2. Analisi dei Ruoli e delle Responsabilità
   Obiettivo: analizzare bene i ruoli emersi nei casi d'uso e porre particolare attenzione nell'attribuzione delle responsabilità a tali ruoli tenendo conto dell'analisi del rischio
3. Scomposizione del problema
   Obiettivo: se possibile suddividere il problema in sotto-problemi più piccoli

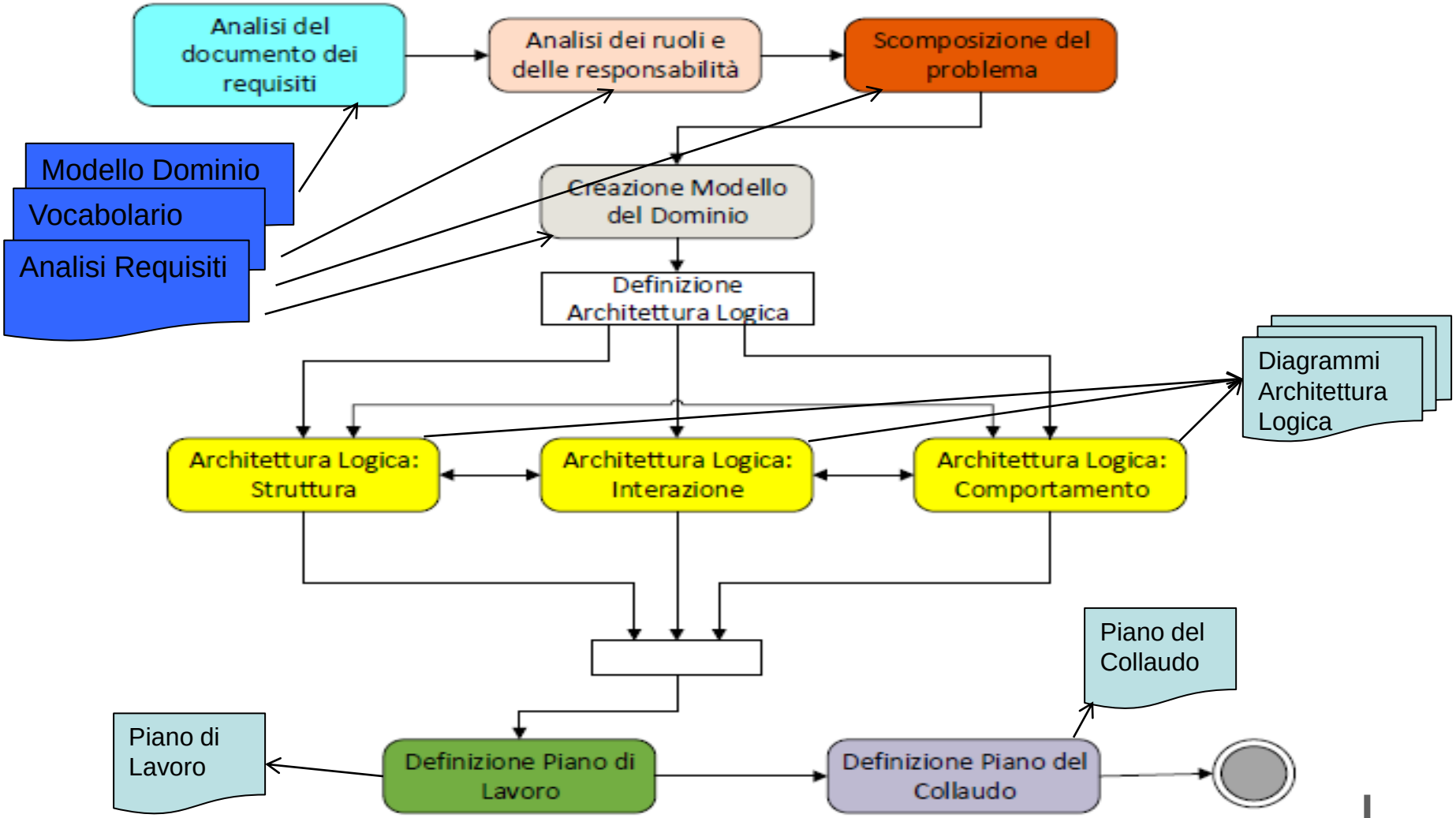
Analisi del Problema

4. Creazione del Modello del Dominio
   Obiettivo: partendo dal vocabolario si costruisce il diagramma delle classi del dominio
5. Architettura Logica: Struttura
   Obiettivo: creazione dei diagrammi strutturali (package e classi) dell'Architettura Logica
6. Architettura Logica: Interazione
   Obiettivo: creazione dei diagrammi di interazione (diagrammi di sequenza) dell'Architettura Logica
7. Architettura Logica: Comportamento
   Obiettivo: creazione dei diagrammi di comportamento (stato e attività) dell'Architettura Logica

Analisi del Problema

8. Definizione del Piano di Lavoro
   Obiettivo: assegnare le responsabilità
   a ciascun membro del team di progetto,
   stabilire i vincoli temporali,
   impostare eventuali milestone, …
9. Definizione del Piano del Collaudo
   Obiettivo: definire i risultati attesi da ogni entità
   (classe, sottosistema) che compare nell'Architettura
   Logica

Analisi del Problema



Architettura Logica

- L'**Architettura Logica** è un insieme di modelli UML costruiti per definire una struttura concettuale del problema robusta e modificabile
- Attraverso l'Architettura Logica l'analista descrive
  - la struttura (insieme delle parti)
  - il comportamento atteso (da tutto e da ciascuna parte)
  - le interazioni

  così come possono essere dedotte dai requisiti, dalle caratteristiche del problema e del dominio applicativo **senza alcun riferimento alle tecnologie realizzative**

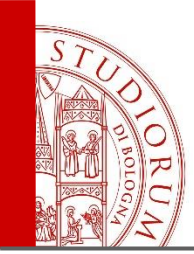
Piano di Lavoro

- Il Piano di Lavoro esprime l'articolazione delle attività in termini di risorse
  - umane
  - temporali
  - di elaborazione
  - …

  necessarie allo sviluppo del prodotto
  in base ai risultati dell'Analisi del Problema

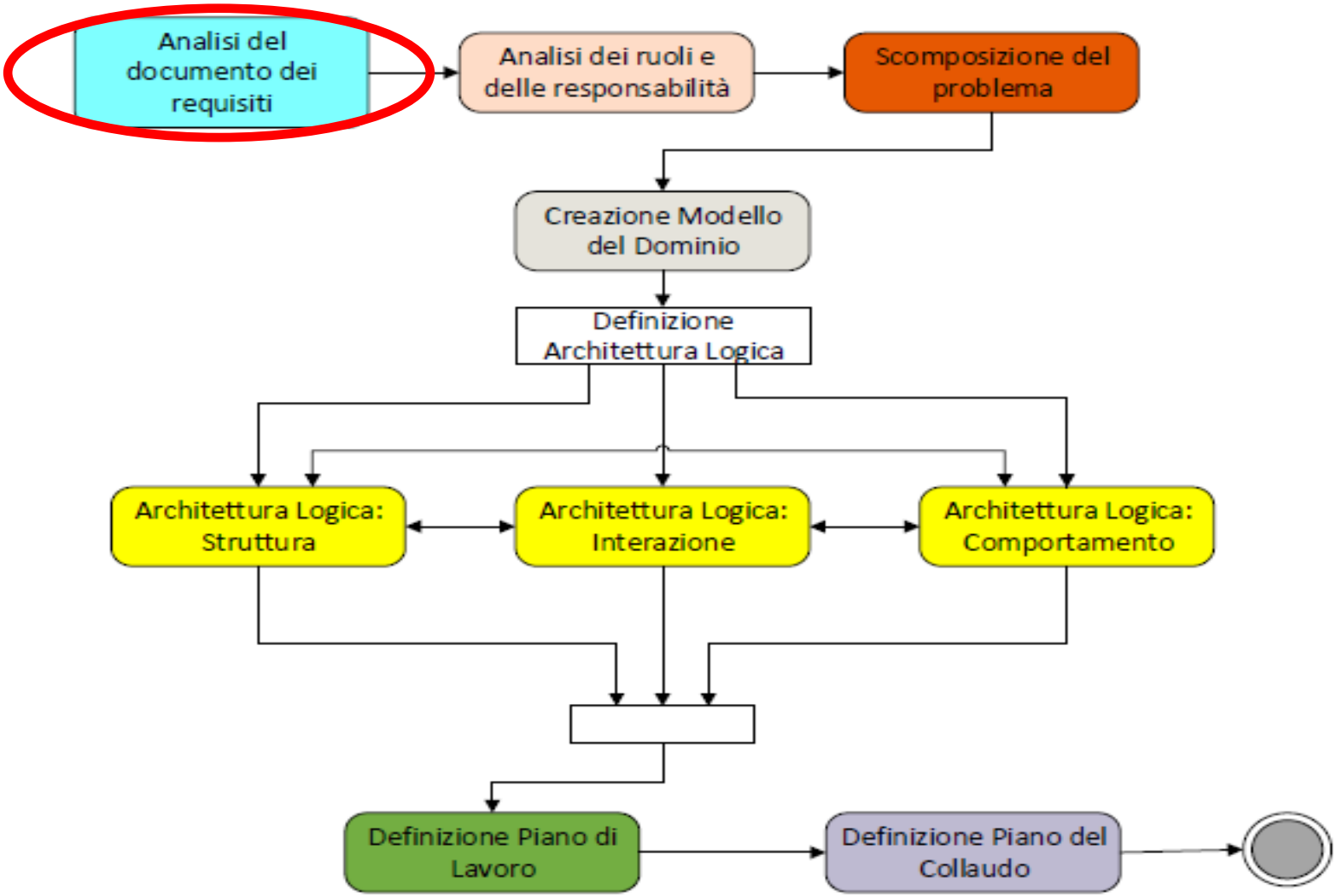
Piano del Collaudo

- Il Piano del Collaudo definisce l'insieme dei risultati attesi
  da ogni entità definita nell'Architettura Logica
  in relazione a specifiche sollecitazioni stimolo-risposta prevista
- Un modo per impostare il Piano del Collaudo
  è quello di fornire i test delle classi definite nell'Architettura Logica,
  così da pianificare già in questa fase i test di integrazione
  dei vari sottosistemi
- Ciò agevola il lavoro della successiva fase di progetto,
  riducendo il rischio di brutte sorprese in fase di integrazione
  delle sotto-parti
- Strumenti tipici per scrivere i test-case:
  - JUnit nel mondo Java
  - NUnit (http://nunit.org/) nel mondo C#



Analisi Documento dei Requisiti

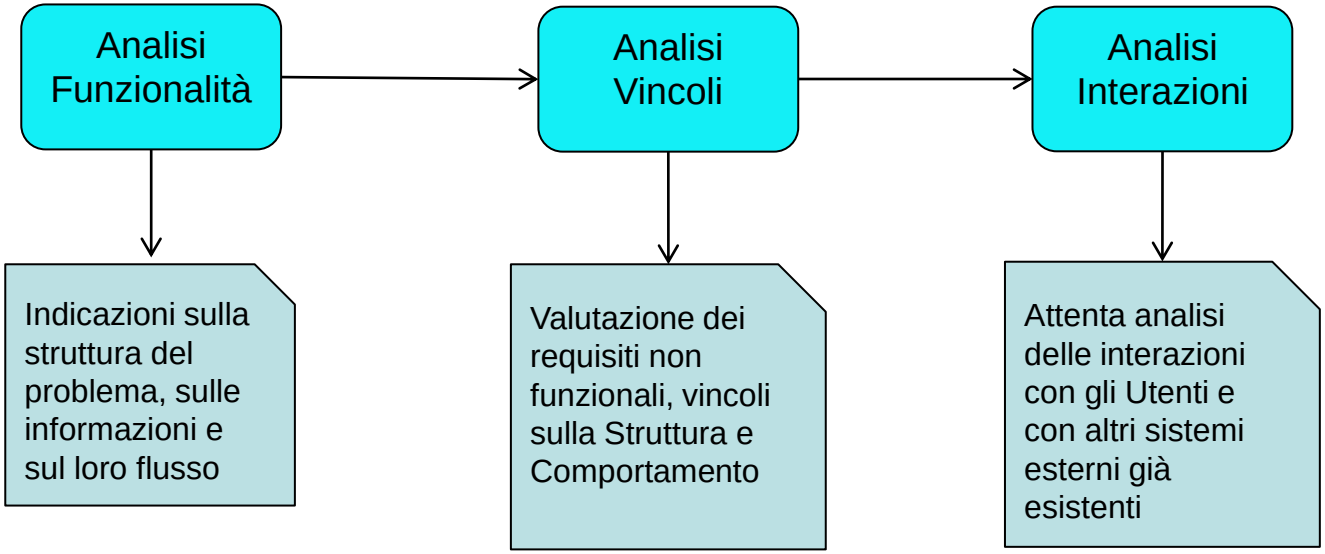
Analisi del Problema



Analisi Documento dei Requisiti

- Il Documento dei Requisiti evidenzia
  - le funzionalità / i servizi che dovranno essere sviluppati (requisiti funzionali)
  - i vincoli che dobbiamo tenere in considerazione (requisiti non funzionali)
  - il "mondo esterno" con cui si dovrà interagire
  - i "rischi" legati a possibili attacchi alla protezione, integrità e privacy dei dati (requisiti di sicurezza)
- Partendo da tale documento occorre applicare un'attenta analisi delle singole funzionalità, vincoli, interazioni e rischi

Analisi Documento dei Requisiti



Analisi delle Funzionalità

- Per ogni funzionalità espressa nei Casi d'Uso vanno analizzati:
  - *il tipo, ovvero l'obiettivo della funzionalità*: memorizzazione dei dati, interazione con l'esterno, gestione/manipolazione dei dati
    - etichettiamo la funzionalità in modo da rendere evidente durante la creazione dell'Architettura Logica dove "posizionarla"
    - se la funzionalità è molto complessa potrebbe appartenere a più tipi differenti: marcare la funzionalità come "complessa"
  - *le informazioni coinvolte*: analisi sistematica di tutte le informazioni sulle quali deve "operare" la funzionalità
    - In particolare individuare il tipo di dato (composto/singolo) e il grado di riservatezza richiesto
  - *il "flusso delle informazioni"*: quali sono le informazioni che
    - si ricevono in input: permette di dare indicazioni sulla validazione dell'input e se sono presenti vincoli sui valori ammessi
    - vanno prodotte in output: permette di valutare dall'esterno se la funzionalità si comporta come ci si aspetta

Analisi delle Funzionalità

- Si possono predisporre diverse tabelle di analisi

Tabella Funzionalità

Funzionalità	Tipo	Grado Complessità	Requisiti Collegati
nome funzionalità come compare nei casi d'uso	tipo della funzionalità	indicare se complessa o semplice	Requisiti a cui è collegata la funzionalità

Tabella Informazioni / Flusso

Informazione	Tipo	Livello di riservatezza/ privacy	Input/Output	Vincoli
nome (o id)	composto/ semplice	grado di riservatezza richiesto	specificare se input o output	eventuali vincoli sui valori attesi

Analisi delle Funzionalità

- Si possono predisporre diverse tabelle di an

Una sola tabella per tutte le funzionalità

Tabella Funzionalità

Funzionalità	Tipo	Grado Complessità	Requisiti Collegati
nome funzionalità	tipo della funzionalità	indicare se complessa o semplice	Requisiti a cui è collegata la funzionalità

Una tabella per ogni funzionalità

Tabella Informazioni / Flusso

Informazione	Tipo	Livello di riservatezza/ privacy	Input/Output	Vincoli
nome (o id)	composto/ semplice	grado di riservatezza richiesto	specificare se input o output	eventuali vincoli sui valori attesi

Esempio

- Tabella Funzionalità per il Villaggio Turistico

Funzionalità	Tipo	Grado Complessità	Requisiti Collegati
GestioneOspite	Memorizzazione dati, gestione dati	complessa	R1F, R2F, R3F, R4F,R5F, R7F, R8F, R10F, R12F, R13F
Login	Interazione esterno, gestione dati	semplice	R14F, R15F
NuovoMovimento	Memorizzazione dati, gestione dati	semplice	R11F
ElencoVendite	Gestione dati	semplice	R6F
ScritturaLog	Memorizzazione dati	semplice	R16F
AnalisiLog	Gestione dati	complessa	R17F

Esempio

- Tabella Informazioni/Flusso per NuovoMovimento

Informazione	Tipo	Livello protezione / privacy	Input/output	Vincoli
Data	semplice	Protezione media	Input	Non più di 10 caratteri
Ora	semplice	Protezione media	Input	Non più di 5 caratteri
Tipo di acquisto	semplice	Protezione alta	Input	Non più di 400 caratteri
Importo addebitato	semplice	Protezione alta	Input	Non più di 30 caratteri
Identificativo GuestCard	semplice	Protezione molto alta	Input	Non più di 20 caratteri
Indentificativo Punto Vendita	semplice	Protezione molto alta	Input	Non più di 20 caratteri
Identificativo Catena Punti di Vendita	semplice	Protezione molto alta	Input	Non più di 20 caratteri

Analisi dei Vincoli

- Per ogni requisito non funzionale vanno analizzati:
  - *il tipo, ovvero a quale categoria appartiene*:
    vincoli sulle performance, sui tempi di risposta, sulla scalabilità, sull'usabilità, sulla protezione dei dati, etc…
    - etichettiamo il requisito in modo da rendere evidente la categoria
    - alcuni requisiti potrebbero influenzare diversi aspetti del sistema, nel caso indicarli tutti, insieme anche al tipo di impatto (es: portano ad un peggioramento)
  - *le funzionalità coinvolte*: indicare le funzionalità che vengono influenzate dal requisito

Analisi dei Vincoli

- Si può predisporre una tabella

Tabella Vincoli

Requisito	Categorie	Impatto	Funzionalità
Requisito non funzionale	tipo	indicare il tipo di impatto	funzionalità coinvolte

Una sola tabella per tutti i vincoli

Esempio

- Tabella Vincoli per il Villaggio Turistico

Requisito	Categorie	Impatto	Funzionalità
Velocità ricerca dati (R1NF)	Tempo di risposta	Cercare di migliorare	GestioneOspite, NuovoMovimento, ElencoVendite, Login
Velocità memorizzazione dati (R3NF)	Tempo di risposta	Cercare di migliorare	GestioneOspite, NuovoMovimento, ElencoVendite
Facile navigabilità delle schermate (R2NF)	Usabilità	Cercare di migliorare	GestioneOspite, NuovoMovimento, ElencoVendite, Login
Protezione dei dati (R3NF, R4NF, R6NF, R7NF, R8NF, R12NF)	Sicurezza	Peggiorano tempo di risposta, migliorano la privacy dei dati	GestioneOspite, NuovoMovimento, ElencoVendite, Login
Controllo Accessi (R5NF, R8NF, R9NF, R11NF)	Sicurezza	Peggiorano tempo di risposta e usabilità, migliorano la privacy dei dati	GestioneOspite, NuovoMovimento, ElencoVendite, Login

Analisi delle Interazioni

- Vanno distinte le interazioni con gli umani
  da quelle con sistemi esterni
- Nel caso delle interazioni con gli umani
  - analizzare le eventuali interfacce identificate con il cliente nella fase di Analisi dei Requisiti, oppure delineare le possibili interfacce
  - individuare le maschere di inserimento/output dati
  - individuare le sole informazioni necessarie da mostrare in ogni maschera
  - creare un legame tra maschere – informazioni – funzionalità

Analisi delle Interazioni

- Si può predisporre una tabella

Tabella Maschere

Maschera	Informazioni	Funzionalità
Nome della maschera	indicare le informazioni gestite nella maschera	funzionalità coinvolte

Una sola tabella per tutte le maschere

Esempio

- Tabella Maschere per il Villaggio Turistico

Maschera	Informazioni	Funzionalità
Home Gestione	Messaggio benvenuto e scelta funzionalità	GestioneOspite
View Registrazione	Nome Ospite, Cognome Ospite, DataNascita, Indirizzo di residenza, Numero di telefono, Estremi del documento di identificazione, numero della stanza	GestioneOspite
View Apertura Credito	GuestCard, Identificativo OspitePagante	GestioneOspite
View Chiusura Credito	GuestCard, Identificativo OspitePagante, saldo	GestioneOspite
View Elenco Acquisti	Elenco dei Movimenti	GestioneOspite
Home PuntoVendita	Messaggio benvenuto e scelta funzionalità	NuovoMovimento
View Inserimento	Data, Ora, Tipo di acquisto, importo addebitato, GuestCard	NuovoMovimento
Home Catena PuntiVendita	Elenco dei report di vendita per ogni PuntoDiVendita	ElencoVendite
Home Log	Scelta del tipo di analisi o di visione di tutto il log di una parte del sistema	AnalisiLog
View Log	Data, Ora, Operazione eseguita, Messaggio	AnalisiLog
View Anomalie	Elenco delle anomalie	AnalisiLog
View Login	Username, password	Login

Analisi delle Interazioni

- Nel caso delle interazioni con sistemi esterni
  - analizzare ai morsetti i sistemi esterni con cui si dovrà interagire
  - individuare i protocolli di interazione con tali sistemi

Tabella Sistemi Esterni

Sistema	Descrizione	Protocollo di Interazione	Livello di Protezione
Nome del sistema	Descrizione del sistema e delle sue principali funzionalità	Specificare il protocollo di interazione richiesto	Specificare il livello di protezione garantito dal sistema

Una sola tabella per tutti i sistemi esterni

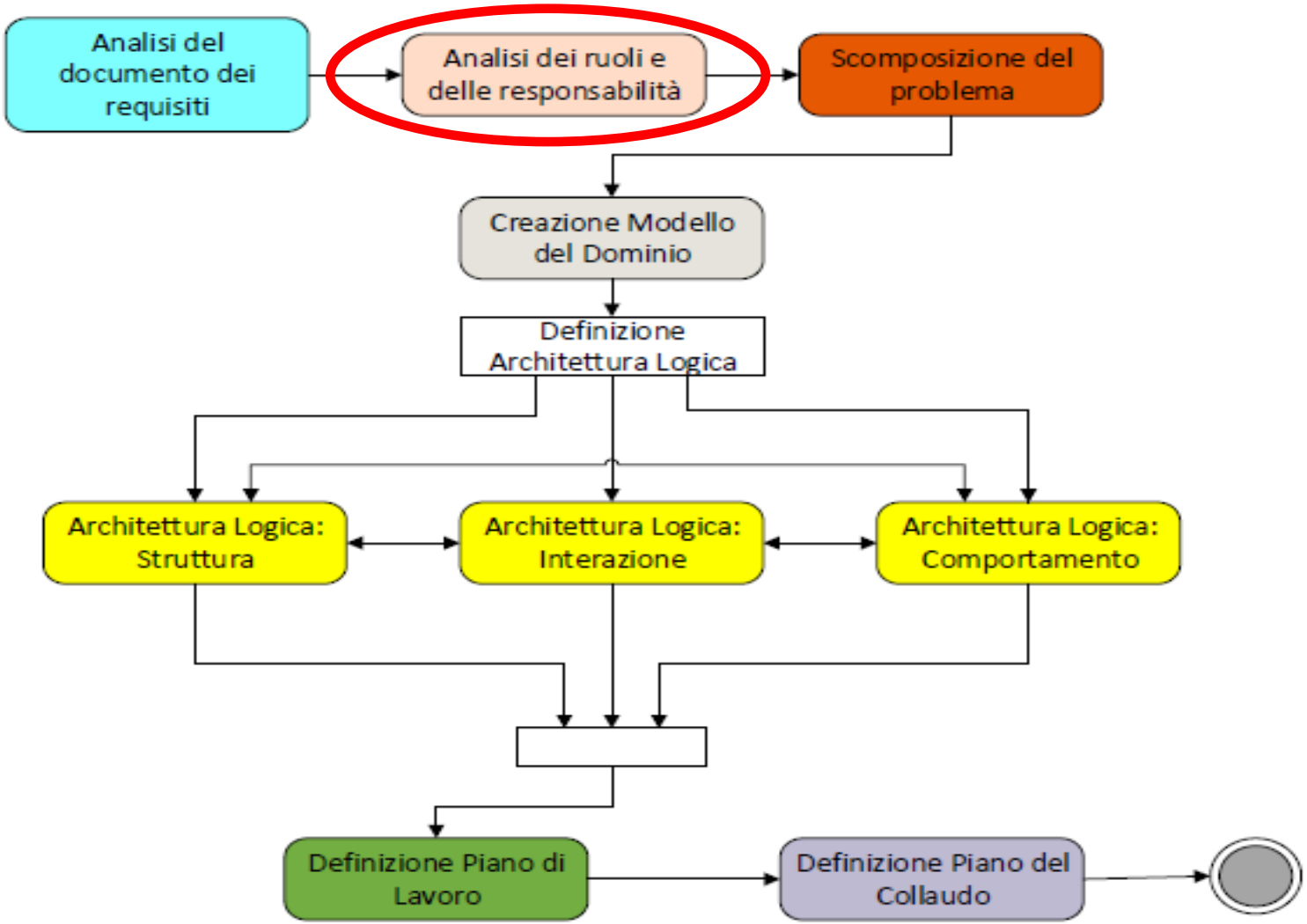
Esempio

- Tabella Sistemi Esterni per il Villaggio Turistico

Sistema	Descrizione	Protocollo di Interazione	Livello di Sicurezza
Gestione Personale (R16F/R9NF)	Sistema che si occupa della gestione del personale che lavora presso il Villaggio Turistico.	GestionePersonale mette a disposizione una funzionalità di controllo delle credenziali. Le credenziali devono essere inviate in modo sicuro e come risultato si ha una stringa che rappresenta il nome del ruolo assegnato al dipendente	Alto livello di sicurezza perché protegge i dati personali e sensibili dei dipendenti

Analisi Ruoli e Responsabilità

Analisi del Problema



Analisi dei Ruoli e Responsabilità

- Per ogni Attore individuato nei Casi d'Uso specificare
  - le responsabilità – cosa deve fare in ogni funzionalità
  - specificare le *informazioni a cui può accedere* relativamente a ciascuna funzionalità in cui è coinvolto con relativa indicazione del tipo di accesso
  - le maschere che può visualizzare
  - il suo livello di riservatezza – quale livello di riservatezza è necessario avere per poter ricoprire quel ruolo
  - la numerosità attesa – il numero di persone che possono giocare quel ruolo

Analisi dei Ruoli e Responsabilità

- Si possono predisporre alcune tabelle

Tabella Ruoli

Ruolo	Responsabilità	Maschere	Riservatezza	Numerosità
Nome del ruolo	Indicare le responsabilità assegnate	Indicare le maschere che devono essere visualizzate	Livello di riservatezza necessaria	Indicare la numerosità massima

Tabella Ruolo-Informazioni

Informazione	Tipo Accesso
Informazione	Specificare il tipo di accesso

Analisi dei Ruoli e Responsabilità

- Si possono predisporre alcune tabelle

Una sola tabella per tutti i ruoli

Tabella Ruoli

Ruolo	Responsabilità	Maschere	Riservatezza	Numerosità
Nome del ruolo	Indicare le responsabilità assegnate	Indicare le maschere che devono essere visualizzate	Livello di riservatezza necessaria	Indicare la numerosità massima

Tabella Ruolo-Informazioni

Una tabella per ogni ruolo

Informazione	Tipo Accesso
Informazione	Specificare il tipo di accesso

Esempio

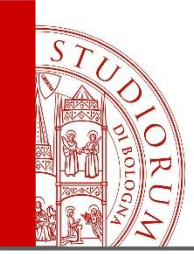
- Tabella Ruoli per il Villaggio Turistico

Ruolo	Responsabilità	Maschere	Riservatezza	Numerosità
Operatore	Gestione di tutte le informazioni relative agli Ospiti del villaggio turistico	Home Gestione, View Registrazione, View Apertura Credito, View Chiusura Credito, View Elenco Acquisti, View Login	È richiesto un alto grado di riservatezza	Massimo 10 Operatori, considerando l'alternanza dei turni di lavoro e dei giorni di risposo
..	…	…	…	…

Esempio

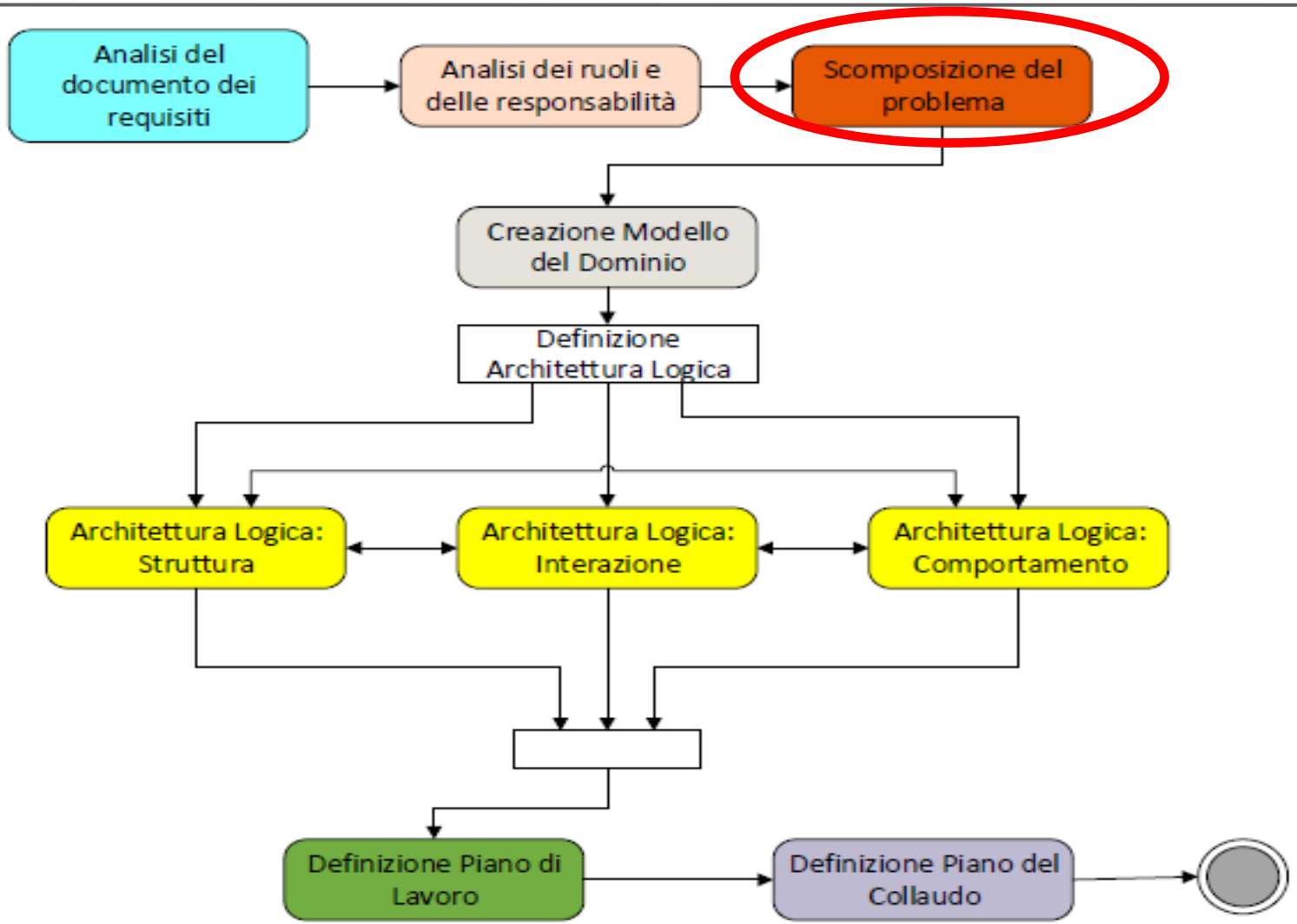
- Tabella Operatore-Informazioni per il Villaggio Turistico

Informazione	Tipo di Accesso
Nome Ospite	Lettura/scrittura
Cognome Ospite	Lettura/scrittura
DataNascita	Lettura/scrittura
Indirizzo di residenza	Lettura/scrittura
Numero di telefono	Lettura/scrittura
Estremi del documento di identificazione utilizzato	Lettura/scrittura
Numero della stanza	Lettura/scrittura
CartaCredito	Scrittura
Identificativo GuestCard	Lettura/scrittura
DataInizioSoggiorno	Lettura/scrittura
DataFineSoggiorno	Lettura/Scrittura
Valuta	Lettura/scrittura
Saldo	Lettura
Username	Scrittura
Password	Scrittura



Scomposizione del Problema

Analisi del Problema



Scomposizione del Problema

- Il punto di partenza è l'Analisi delle Funzionalità
- Per ogni funzionalità marcata come "*complessa*" nella Tabella delle Funzionalità occorre valutare se sia possibile operare una scomposizione
  - la funzionalità può essere suddivisa in sotto-funzionalità più semplici?
  - quale "legame" sussiste tra le sotto-funzionalità?
  - quali informazioni devono "fluire" tra le sotto-funzionalità

Scomposizione del Problema

- Si possono predisporre alcune tabelle

Tabella Scomposizione Funzionalità

Funzionalità	Scomposizione
nome funzionalità	elenco delle sotto-funzionalità

Tabella Sotto-Funzionalità

Sotto-Funzionalità	Sotto-Funzionalità	Legame	Informazioni
nome sotto-funzionalità	nome sotto-funzionalità	specificare il tipo di dipendenza/legame logico	specificare le informazioni scambiate

Scomposizione del Problema

- Si possono predisporre alcune tabelle

Una sola tabella per tutte le funzionalità

Tabella Scomposizione Funzionalità

Funzionalità	Scomposizione
nome funzionalità	elenco delle sotto-funzionalità

Una tabella per ogni funzionalità

Tabella Sotto-Funzionalità

Sotto-Funzionalità	Sotto-Funzionalità	Legame	Informazioni
nome sotto-funzionalità	nome sotto-funzionalità	specificare il tipo di dipendenza/legame logico	specificare le informazioni scambiate

Esempio

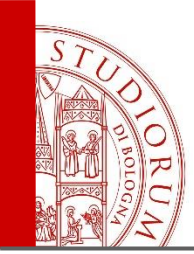
- Tabella Scomposizione Funzionalità per il Villaggio Turistico

Funzionalità	Scomposizione
GestioneOspite	Registrazione, AperturaCredito, ChiusuraCredito, ElencoAcquisti

Esempio

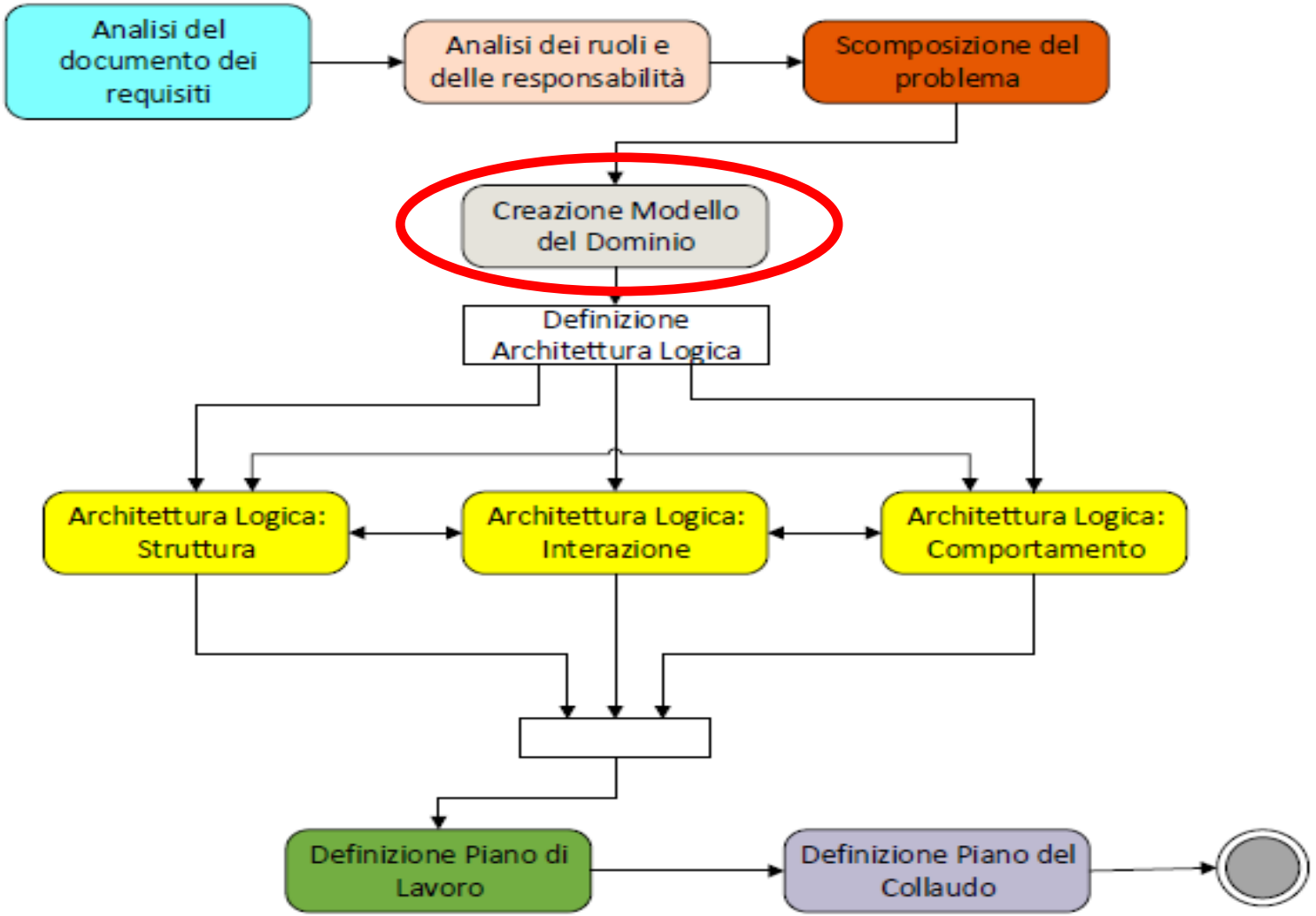
- Tabella Sotto-Funzionalità per il Villaggio Turistico

Sotto-funzionalità	Sotto-funzionalità	Legame	Informazioni
AperturaCredito	Registrazione	AperturaCredito dipende da Registrazione	Identificativo OspitePagante
ChiusuraCredito	AperturaCredito	Non si può chiudere un credito se non è mai stato aperto	Identificativo OspitePagante
ElencoAcquisti	AperturaCredito	Non si può mostrare l'elenco acquisti se non è stato aperto prima il credito	Identificativo OspitePagante



Creazione Modello del Dominio

Analisi del Problema



Creazione Modello del Dominio

- Il punto di partenza è costituito dall'insieme di:
  - glossario definito nell'Analisi dei Requisiti
  - tabelle informazioni/flusso
- Sulla base di questi si costruisce il diagramma delle classi che rappresenta il Modello del Dominio
- Tale modello poi sarà riusato nell'Architettura Logica come modello dei dati

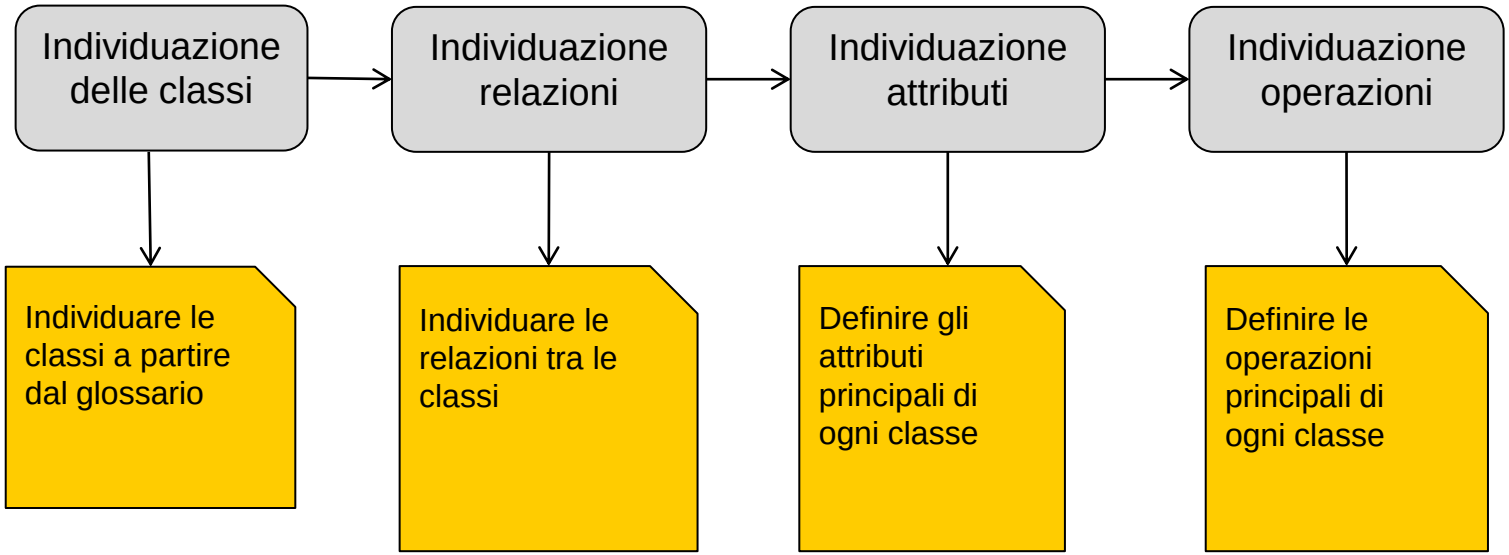
Creazione Modello del Dominio

- Occorre tenere presente che non tutti i vocaboli elencati nel glossario diventeranno classi, si devono infatti evitare:
  - ridondanze: classi uguali ma con diverso nome
  - costruzione di classi a partire da termini ambigui nel glossario
  - nomi che si riferiscono a eventi o operazioni
  - nomi appartenenti al meta-linguaggio del processo come, per esempio, *requisiti e sistema*
  - nomi al di fuori dell'ambito del sistema
  - nomi che possono essere attributi

Modello del Dominio

- Individuare
  - Oggetti e classi rilevanti per il problema che si sta analizzando
    - Limitarsi esclusivamente a quelle classi che fanno parte del vocabolario del dominio del problema
  - Relazioni tra le classi
  - Per ogni classe
    - Attributi
    - Operazioni fondamentali cioè servizi forniti all'esterno

Modello del Dominio



Individuazione delle Classi

- Dal glossario eliminare i nomi che sicuramente
  - non si riferiscono a classi
  - indicano attributi (dati di tipo primitivo)
  - indicano operazioni
- Scegliere un solo termine significativo
  se più parole indicano lo stesso concetto (sinonimi)
- Il nome della classe deve essere un nome familiare
  - all'utente o all'esperto del dominio del problema
  - non allo sviluppatore!

Individuazione delle Classi

- Attenzione agli aggettivi e agli attributi, possono
  - indicare oggetti diversi
  - indicare usi diversi dello stesso oggetto
  - essere irrilevanti
- Ad esempio:
  - “Studente bravo” potrebbe essere irrilevante
  - “Studente fuori corso” potrebbe essere una nuova classe
- Attenzione alle frasi passive, impersonali o con soggetti fuori dal sistema:
  - devono essere rese attive ed esplicite, perché potrebbero mascherare entità rilevanti per il sistema in esame

Individuazione delle Classi

- Individuare Attori con cui il sistema in esame deve interagire
  - Persone: Docente, Studente, Esaminatore, Esaminando, …
  - Sistemi esterni: ReteLocale, Internet, DBMS, …
  - Dispositivi: attuatori, sensori, …
- Individuare modelli e loro elementi specifici:
  - Insegnamento – “Ingegneria del Software T”
  - CorsoDiStudio – “Ingegneria Informatica”
  - Facoltà – “Ingegneria”
- Individuare cose tangibili, cioè oggetti reali appartenenti al dominio del problema
  - Banco, LavagnaLuminosa, Schermo, Computer, …

Individuazione delle Classi

- Individuare contenitori (fisici o logici) di altri oggetti
  - Facoltà, Dipartimento, Aula, SalaTerminali, …
  - ListaEsame, CommissioneDiLaurea, OrdineDegliStudi, …
- Individuare eventi o transazioni che il sistema deve gestire e memorizzare
  - possono avvenire in un certo istante (es., una vendita) o
  - possono durare un intervallo di tempo (es., un affitto)
  - Appello, EsameScritto, Registrazione, AppelloDiLaurea, …

Individuazione delle Classi

- Per determinare se includere una classe nel modello, porsi le seguenti domande:
  - il sistema deve interagire in qualche modo con gli oggetti della classe?
    - utilizzare informazioni (attributi) contenute negli oggetti della classe
    - utilizzare servizi (operazioni) offerti dagli oggetti della classe
  - quali sono le responsabilità della classe nel contesto del sistema?

Individuazione delle Classi

- Attributi e operazioni devono essere applicabili a tutti gli oggetti della classe
- Se esistono
  - attributi con un valore ben definito solo per alcuni oggetti della classe
  - operazioni applicabili solo ad alcuni oggetti della classe

  siamo in presenza di ereditarietà
- Esempio: dopo una prima analisi, la classe Studente potrebbe contenere un attributo *booleano* inCorso, ma un'analisi più attenta potrebbe portare alla luce la gerarchia:
  - Studente: StudenteInCorso, StudenteFuoriCorso

Individuazione delle Classi:VT

Voce	Definizione	Sinonimi
Villaggio Turistico	Luogo dove si effettua una vacanza	
Ospite	Persona che è in vacanza nel Villaggio Turistico, senza fare distinzione se Pagante o non Pagante	Cliente
OspitePagante	Persona in vacanza nel villaggio turistisco a cui sono associate una o più stanze. Persona che effettua il saldo del conto	Cliente
OspiteNonPagante	Persona in vacanza nel villaggio turistico a cui è associata una stanza. Non effettua pagamenti ed è sempre associato a un OspitePagante	
OspiteNonPagante Maggiorenne	Persona in vacanza nel villaggio turistico a cui è associata una stanza. Ha raggiunto la maggiore età. Gli viene consegnata una GuestCard, ma questa viene associata all'OspitePagante.	
OspiteNonPagante Minorenne	Persona in vacanza nel villaggio turistico a cui è associata una stanza. Non ha raggiunto la maggiore età.	
Stanza	Ambiente fisico in cui gli Ospiti dormono e tengono i loro beni. Ha un certo numero di posti disponibili	camera
StanzaCollegata	Ambiente fisico in cui gli Ospiti dormono e tengono i loro beni. È logicamente collegata ad un'altra stanza	camera

Individuazione delle Classi:VT

Ospite

OspitePagante

OspiteNonPagante

OspiteNonPagante Minorenne

OspiteNonPagante Maggiorenne

Stanza

StanzaCollegata (?)

Individuazione delle Relazioni

- La maggior parte delle classi (degli oggetti) interagisce con altre classi (altri oggetti) in vari modi
- In ogni tipo di relazione, esiste un cliente C che **dipende** da un fornitore di servizi F
  - C ha bisogno di F per lo svolgimento di alcune funzionalità che C non è in grado di effettuare autonomamente
  - Conseguenza
    per il corretto funzionamento di C
    è indispensabile il corretto funzionamento di F

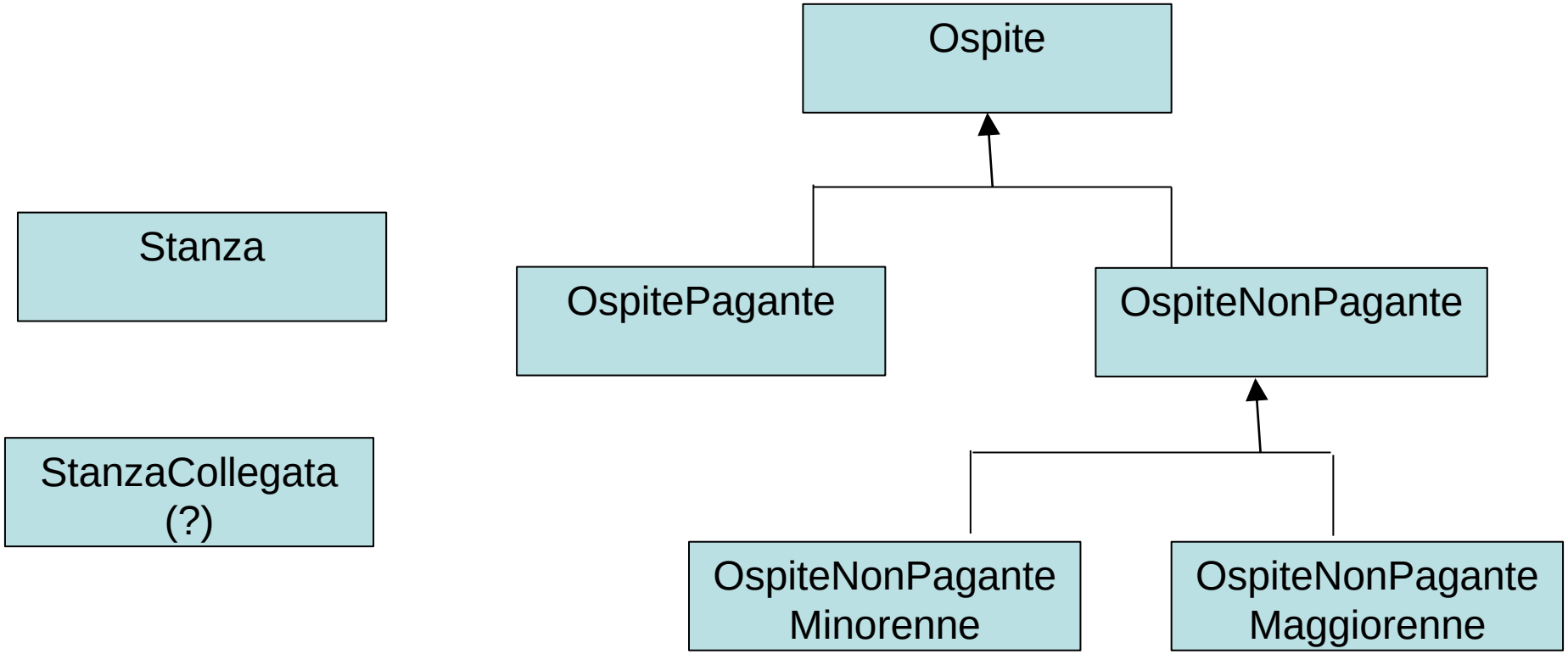
Individuazione delle Relazioni

Tipo di relazione	Cliente	Fornitore
Ereditarietà	sottoclasse	superclasse
Associazione (composizione / aggregazione)	contenitore	contenuto
Dipendenza	classe dipendente (source)	classe da cui si dipende (target)

Individuazione dell'Ereditarietà

- L'ereditarietà deve rispecchiare una tassonomia effettivamente presente nel dominio del problema
  - Non usare l'ereditarietà dell'implementazione (siamo ancora in fase di analisi!)
  - Non usare l'ereditarietà solo per riunire caratteristiche comuni
    - ad es., Studente e Dipartimento hanno entrambi un indirizzo, ma non per questo c'è ereditarietà!

Ereditarietà:VT



Individuazione delle Associazioni

- Un'associazione rappresenta una relazione strutturale tra due istanze di classi diverse o della stessa classe
- Un'associazione può
  - Rappresentare un contenimento logico (aggregazione)
    - Una lista d'esame contiene degli studenti
  - Rappresentare un contenimento fisico (composizione)
    - Un triangolo contiene tre vertici
  - Non rappresentare un reale contenimento
    - Una fattura si riferisce a un cliente
    - Un evento è legato a un dispositivo

Individuazione delle Associazioni

- Aggregazione
  Un oggetto x di classe X è associato a (contiene)
  un oggetto y di classe Y in modo non esclusivo
  x può condividere y con altri oggetti anche di tipo diverso
  (che a loro volta possono contenere y)
- Una lista d'esame contiene degli studenti
  - Uno studente può essere contemporaneamente
    in più liste d'esame
  - La cancellazione della lista d'esame non comporta
    l'eliminazione "*fisica*" degli studenti in lista

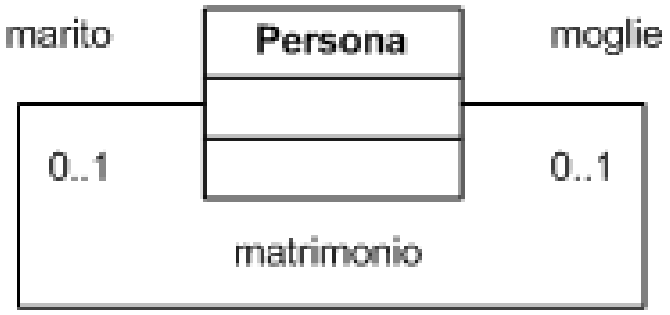
Individuazione delle Associazioni

- Composizione
  Un oggetto x di classe X è associato a (contiene)
  un oggetto y di classe Y in modo esclusivo
  y esiste solo in quanto contenuto in x
- Un triangolo contiene tre punti (i suoi vertici)
  - L'eliminazione del triangolo comporta l'eliminazione dei tre punti
- Se la distruzione del contenitore comporta la distruzione degli oggetti contenuti, si tratta di composizione, altrimenti si tratta di aggregazione

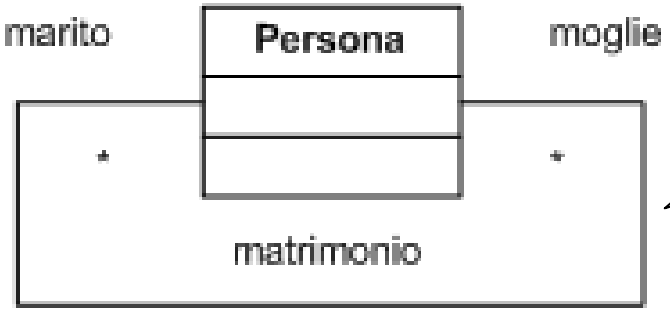
Individuazione delle Associazioni

- Attenzione alle associazioni molti a molti
  possono nascondere una classe
  (**classe di associazione**) del tipo “evento da ricordare”
- Ad esempio,
  - la connessione “matrimonio” tra Persona e Persona
    può nascondere una classe Matrimonio, che lega due Persone
  - la connessione “possiede” tra Proprietario e Veicolo
    può nascondere una classe CompraVendita,
    che lega un Proprietario a un Veicolo

1° Esempio di Associazione

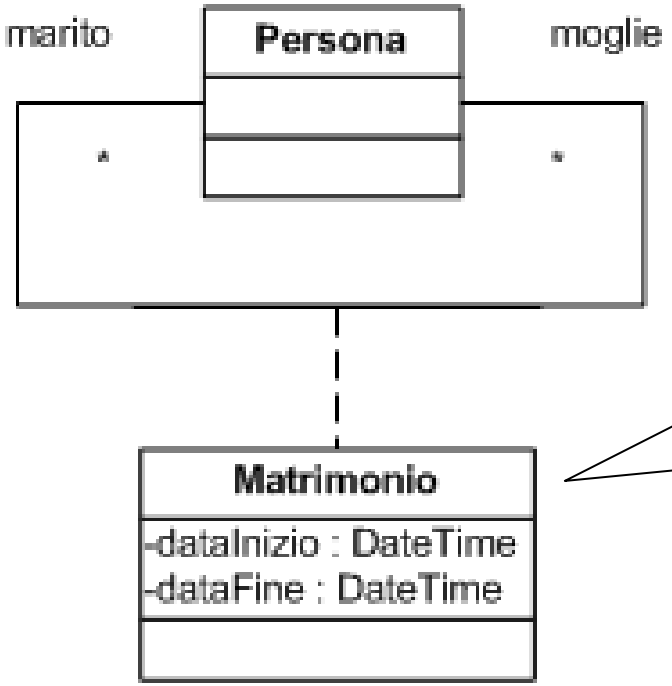


Modella una situazione **corrente**
in cui una Persona è associata
al massimo a un'altra Persona
mediante matrimonio



Modella una situazione **storica**
in cui si tiene traccia
di tutte le associazioni mediante
matrimonio di una Persona
a 0+ altre Persone

1° Esempio di Associazione

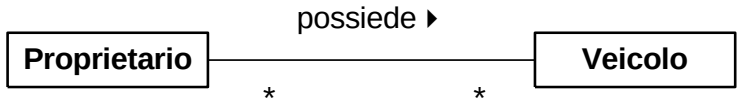


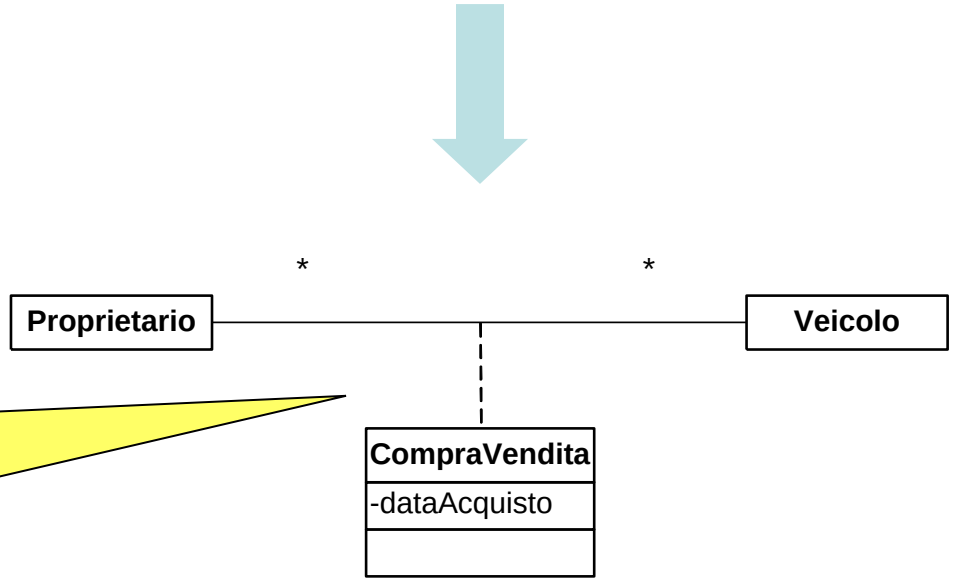
Modella una situazione **storica** in cui si tiene traccia di tutte le associazioni mediante matrimonio di una Persona a 0+ altre Persone

NB: vincoli sulle date

2° Esempio di Associazione

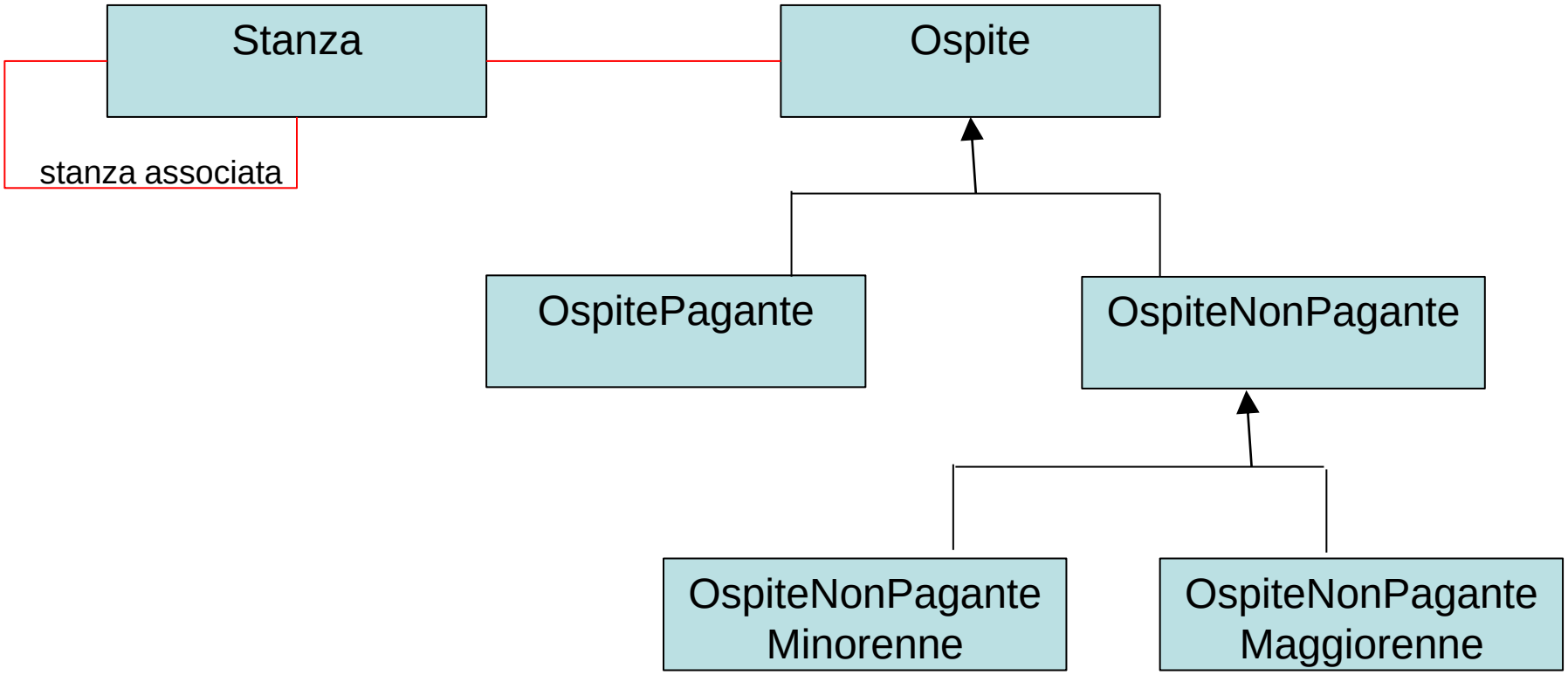
- Un proprietario può possedere molti veicoli
- Un veicolo può essere di molti proprietari
  - in tempi successivi
  - in comproprietà



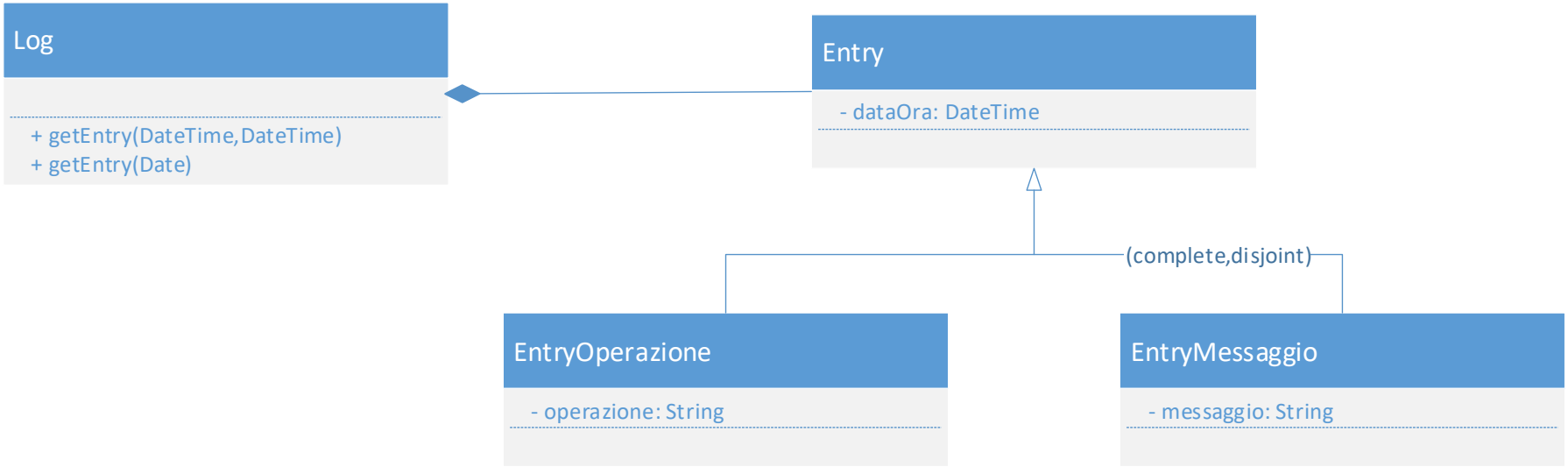


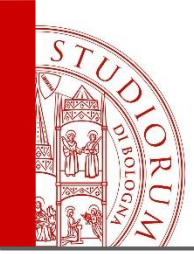
Attenzione alle molteplicità: nel modello delle classi di analisi va bene, ma a livello di istanza si introduce il vincolo extra che ci può essere solo un'istanza della classe di associazione tra ogni coppia di oggetti associati

Associazioni: VT



Associazioni: VT



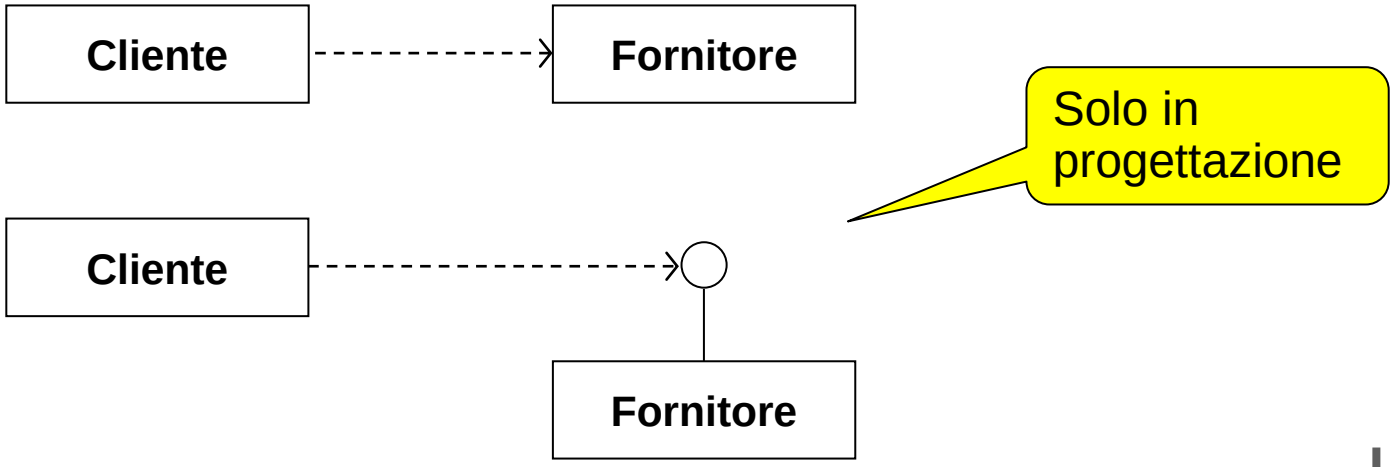


Individuazione Collaborazioni

- Una classe A è in relazione USA con una classe B (A USA B) quando A ha bisogno della collaborazione di B per lo svolgimento di alcune funzionalità che A non è in grado di effettuare autonomamente
  - Un'operazione della classe A ha bisogno come argomento di un'istanza della classe B
    - void fun1(B b) { … usa b … }
  - Un'operazione della classe A restituisce un valore di tipo B
    - B fun2(…) { B b; … return b; }
  - Un'operazione della classe A utilizza un'istanza della classe B (ma non esiste un'associazione tra le due classi)
    - void fun3(…) { B b = new B(…); … usa b … }

Individuazione Collaborazioni

- La relazione non è simmetrica
  A dipende da B, ma B non dipende da A
- Evitare situazioni in cui una classe, tramite una catena di relazioni USA, alla fine dipende da se stessa



Individuazione degli Attributi

- Ogni attributo modella una proprietà atomica di una classe
  - Un valore singolo
    - Una descrizione, un importo, ..
  - Un gruppo di valori strettamente collegati tra loro
    - Un indirizzo, una data, ..
- Proprietà non atomiche di una classe devono essere modellate come associazioni
- A tempo di esecuzione, in un certo istante, ogni oggetto avrà un valore specifico per ogni attributo della classe di appartenenza: informazione di stato

Individuazione degli Attributi

- La base di partenza per la ricerca degli attributi sono le Tabelle di Informazione/Flusso
- Il nome dell'attributo
  - deve essere un nome familiare
    - all'utente o all'esperto del dominio del problema
    - non allo sviluppatore!
  - non deve essere il nome di un valore ("qualifica" sì, "ricercatore" no)
  - *deve iniziare con una lettera minuscola*
- Esempi
  - cognome, dataDiNascita, annoDiImmatricolazione

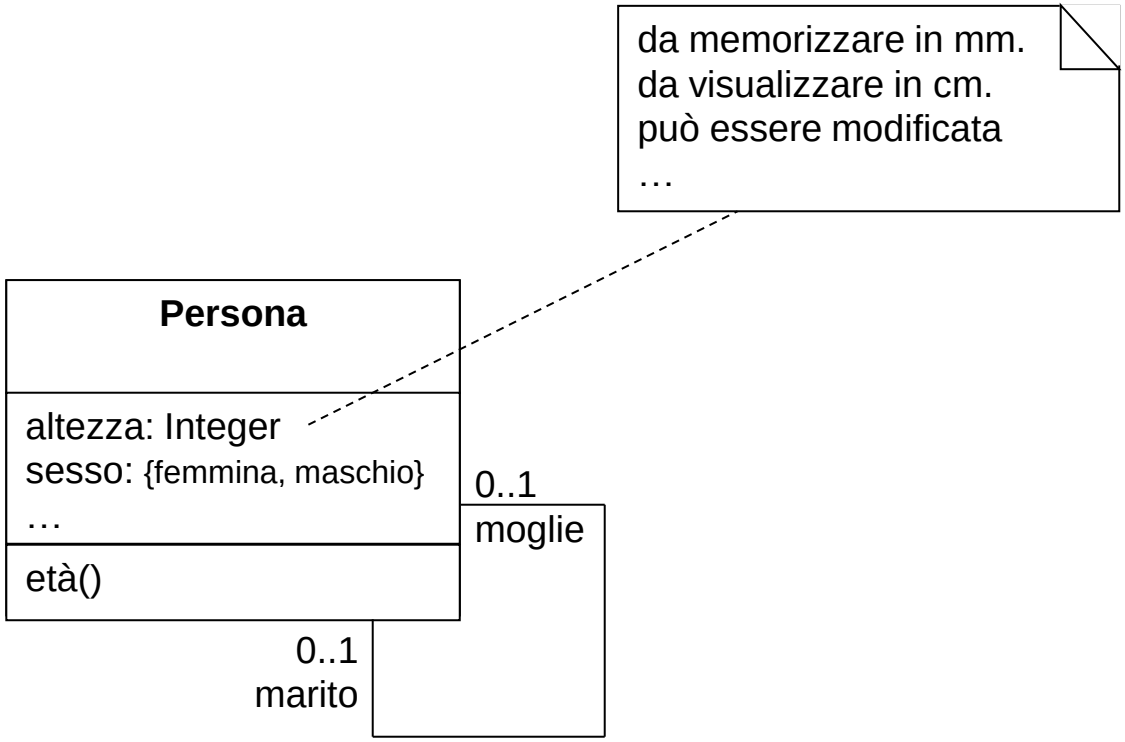
Individuazione degli Attributi

- Esprimere tutti i vincoli applicabili all'attributo
  - Tipo (semplice o enumerativo)
  - Opzionalità
  - Valori ammessi
    - dominio, anti-dominio, univocità
  - Vincoli di creazione
    - valore iniziale di default, immodificabilità del valore (readonly), etc.
  - Vincoli dovuti ai valori di altri attributi
  - Unità di misura, precisione

Individuazione degli Attributi

- Esprimere tutti i vincoli applicabili all'attributo
  - Visibilità (opzionale in fase di analisi)
    Attenzione: gli attributi membro devono essere sempre privati!
  - Appartenenza alla classe (e non all'istanza)
    Attributi (e associazioni) possono essere di classe, cioè essere unici nella classe
    -numIstanze: int = 0
- I vincoli possono essere scritti
  - Utilizzando direttamente UML
  - Utilizzando *Object Constraint Language* (OCL)
  - Come testo in formato libero in un commento UML

Individuazione degli Attributi

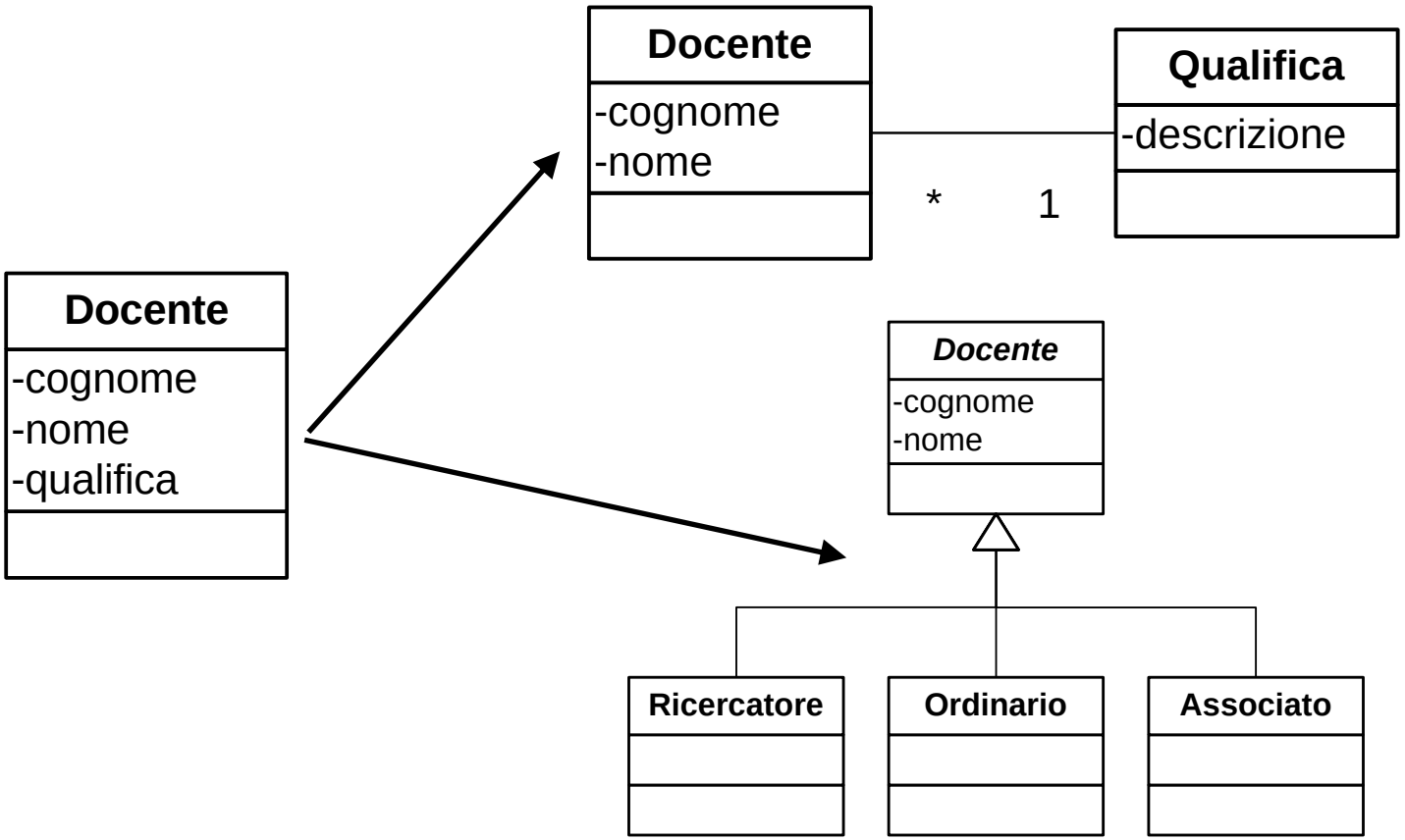


Individuazione degli Attributi

- Attenzione: nel caso di attributi con valore booleano "vero o falso", "sì o no", il nome dell'attributo potrebbe essere uno dei valori di un'enumerazione
  - Ad esempio: tassabile (sì o no) potrebbe diventare tipoTassazione { tassabile, esente, ridotto, … }
- Attenzione: attributi
  - con valore "non applicabile" o
  - con valore opzionale o
  - a valori multipli (enumerazioni)

  possono nascondere ereditarietà o una nuova classe

Individuazione degli Attributi



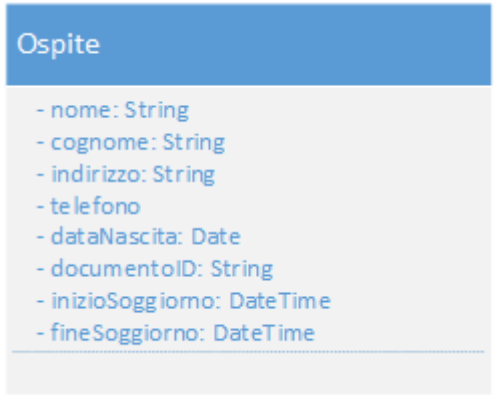
Individuazione degli Attributi

- Attenzione: nel caso di attributi calcolabili (ad esempio, età), specificare sempre l'operazione di calcolo mai l'attributo
  - se memorizzare oppure no un attributo calcolabile è una decisione progettuale, un compromesso tra tempo di calcolo e spazio di memoria
- Applicare l'ereditarietà:
  - Posizionare attributi e associazioni più generali più in alto possibile nella gerarchia
  - Posizionare attributi e associazioni specializzati più in basso

Individuazione degli Attributi: VT

Informazione
Nome Ospite
Cognome Ospite
DataNascita
Indirizzo di residenza
Numero di telefono
Estremi del documento di identificazione utilizzato
CartaCredito
Numero della stanza
Data inizio soggiorno
Data fine soggiorno
Identificativo GuestCard
Movimento
Valuta
Saldo

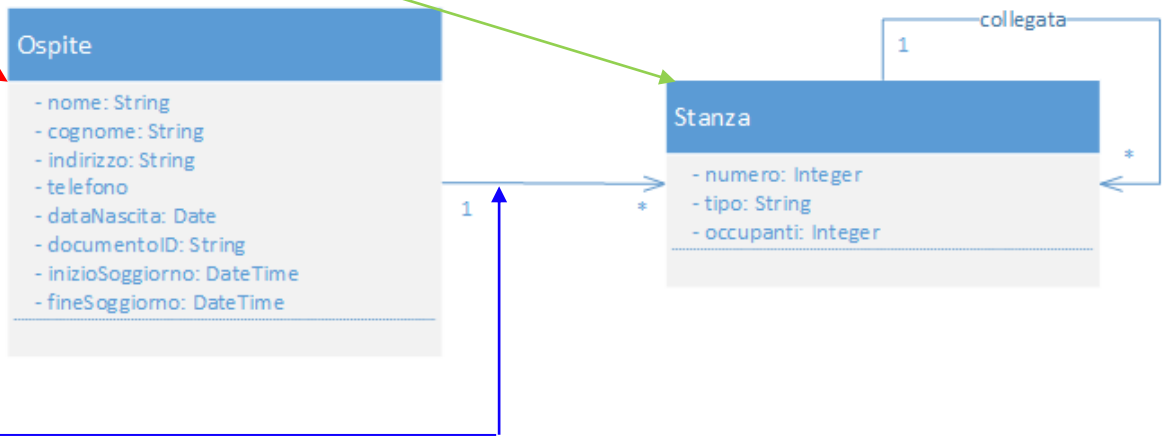
Voce	Definizione
Stanza	Ambiente fisico in cui gli Ospiti dormono e tengono i loro bene. Ha un certo numero di posti disponibili
StanzaCollegata	Ambiente fisico in cui gli Ospiti dormono e tengono i loro bene. È logicamente collegata ad un'altra stanza



Individuazione degli Attributi: VT

Informazione
Nome Ospite
Cognome Ospite
DataNascita
Indirizzo di residenza
Numero di telefono
Estremi del documento di identificazione utilizzato
CartaCredito
Numero della stanza
Data inizio soggiorno
Data fine soggiorno
Identificativo GuestCard
Movimento
Valuta
Saldo

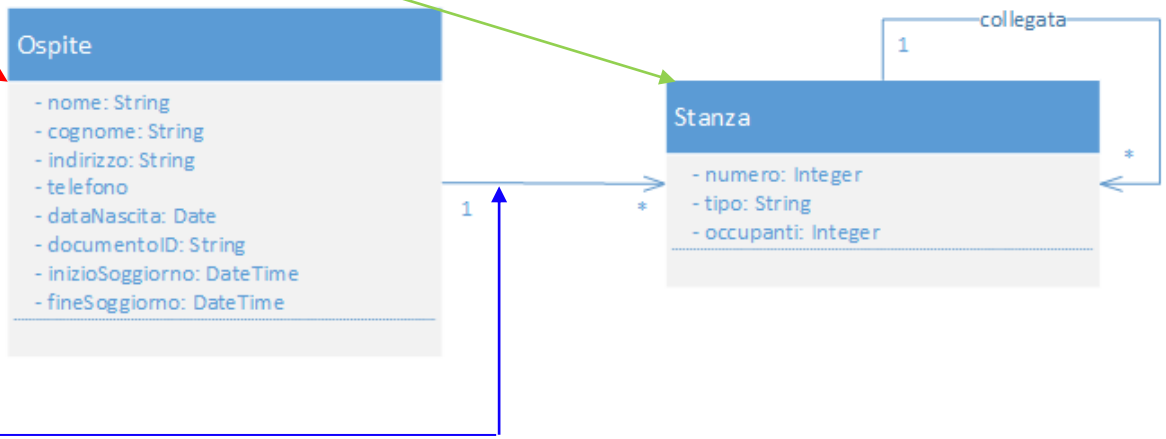
Voce	Definizione
Stanza	Ambiente fisico in cui gli Ospiti dormono e tengono i loro bene. Ha un certo numero di posti disponibili
StanzaCollegata	Ambiente fisico in cui gli Ospiti dormono e tengono i loro bene. È logicamente collegata ad un'altra stanza



Individuazione degli Attributi: VT

Informazione
Nome Ospite
Cognome Ospite
DataNascita
Indirizzo di residenza
Numero di telefono
Estremi del documento di identificazione utilizzato
CartaCredito
Numero della stanza
Data inizio soggiorno
Data fine soggiorno
Identificativo GuestCard
Movimento
Valuta
Saldo

Voce	Definizione
Stanza	Ambiente fisico in cui gli Ospiti dormono e tengono i loro bene. Ha un certo numero di posti disponibili
StanzaCollegata	Ambiente fisico in cui gli Ospiti dormono e tengono i loro bene. È logicamente collegata ad un'altra stanza

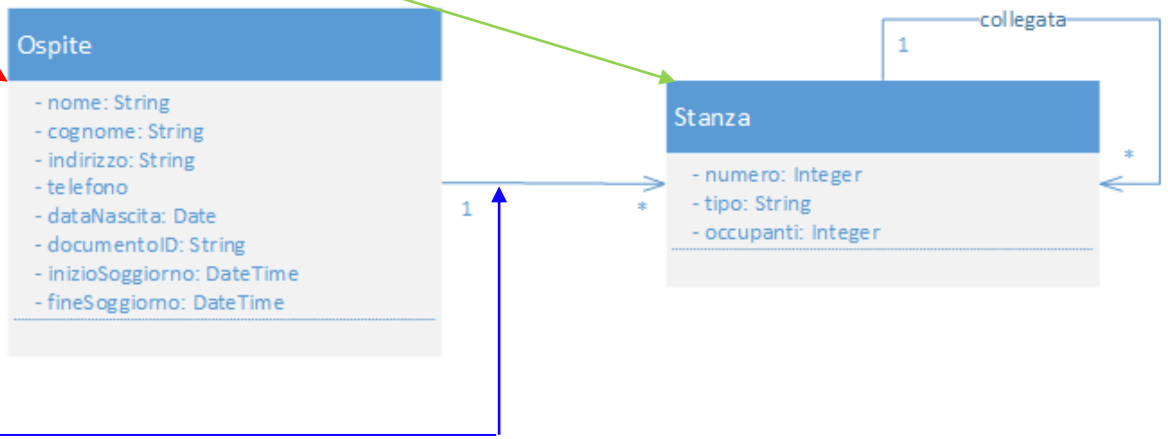


GuestCard è una classe, l'identificativo sarà mappato con una associazione

Individuazione degli Attributi: VT

Informazione
Nome Ospite
Cognome Ospite
DataNascita
Indirizzo di residenza
Numero di telefono
Estremi del documento di identificazione utilizzato
CartaCredito
Numero della stanza
Data inizio soggiorno
Data fine soggiorno
Identificativo GuestCard
Movimento
Valuta
Saldo

Voce	Definizione
Stanza	Ambiente fisico in cui gli Ospiti dormono e tengono i loro bene. Ha un certo numero di posti disponibili
StanzaCollegata	Ambiente fisico in cui gli Ospiti dormono e tengono i loro bene. È logicamente collegata ad un'altra stanza

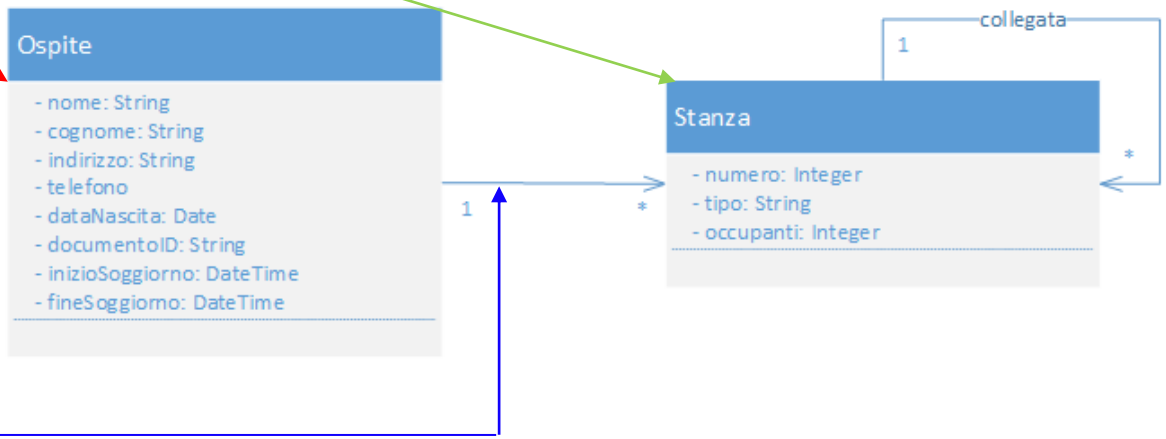


Saldo nasconde una nuova classe, associata alle funzionalità di Apertura e Chiusura Credito → L'ospite ha un conto aperto presso il VT

Individuazione degli Attributi: VT

Informazione
Nome Ospite
Cognome Ospite
DataNascita
Indirizzo di residenza
Numero di telefono
Estremi del documento di identificazione utilizzato
CartaCredito
Numero della stanza
Data inizio soggiorno
Data fine soggiorno
Identificativo GuestCard
Movimento
Valuta
Saldo

Voce	Definizione
Stanza	Ambiente fisico in cui gli Ospiti dormono e tengono i loro bene. Ha un certo numero di posti disponibili
StanzaCollegata	Ambiente fisico in cui gli Ospiti dormono e tengono i loro bene. È logicamente collegata ad un'altra stanza



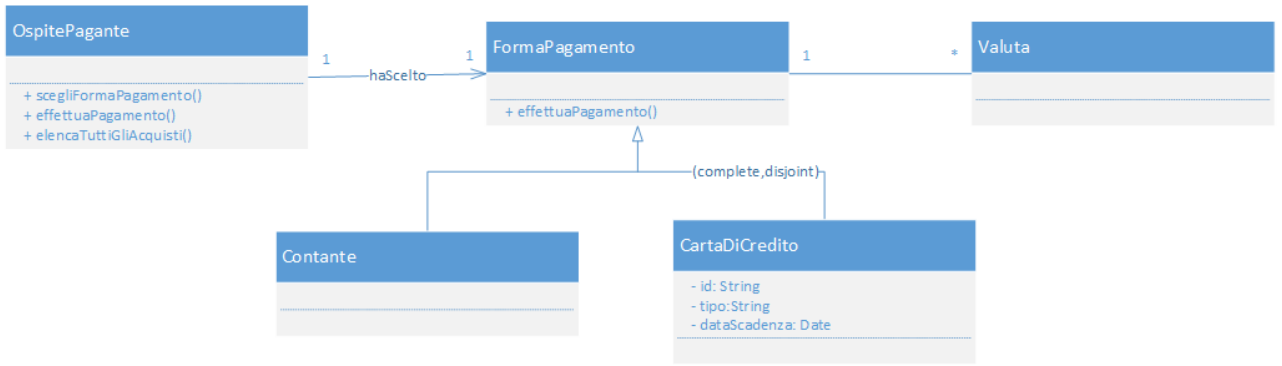
Movimento è relativo all'uso della GuestCard

Individuazione degli Attributi: VT

Informazione
Nome Ospite
Cognome Ospite
DataNascita
Indirizzo di residenza
Numero di telefono
Estremi del documento di identificazione utilizzato
CartaCredito
Numero della stanza
Data inizio soggiorno
Data fine soggiorno
Identificativo GuestCard
Movimento
Valuta
Saldo

Voce	Definizione
Stanza	Ambiente fisico in cui gli Ospiti dormono e tengono i loro bene. Ha un certo numero di posti disponibili
StanzaCollegata	Ambiente fisico in cui gli Ospiti dormono e tengono i loro bene. È logicamente collegata ad un'altra stanza

Carta di Credito e Valuta sono relative alle funzionalità di pagamento scelta e vanno modellate con delle classi



Individuazione delle Operazioni

- Il nome dell'operazione
  - deve appartenere al vocabolario standard del dominio del problema
  - potrebbe essere un verbo
    - all'imperativo (scrivi, esegui, ...) o
    - in terza persona (scrive, esegue, ...)
  - in modo consistente in tutto il sistema

Individuazione delle Operazioni

- Operazioni standard
  - Operazioni che tutti gli oggetti hanno per il semplice fatto di esistere e di avere degli attributi e delle relazioni con altri oggetti (costruttore, accessori, …)
  - Sono implicite e, di norma, non compaiono nel diagramma delle classi di analisi
- Altre operazioni
  - Devono essere determinate
    - servizi offerti agli altri oggetti
  - Compaiono nel diagramma delle classi di analisi

Individuazione delle Operazioni

- Classi contenitori
  - Operazioni standard – aggiungi, rimuovi, conta, itera, …
  - Altre operazioni – riguardano l'insieme degli oggetti, non il singolo oggetto
    - Calcoli da effettuare sugli oggetti contenuti
      Es: calcolaSulleParti(), totalizza()
    - Selezioni da fare sugli oggetti contenuti
      Es: trovaPartiSpecifiche()
    - Operazioni del tipo
      Es: eseguiUnAzioneSuTutteLeParti()

Individuazione delle Operazioni

- Distribuire in modo bilanciato le operazioni nel sistema
- Mettere ogni operazione "*vicino*" ai dati a essa necessari
- Applicare l'ereditarietà
  - Posizionare le operazioni più generali più in alto possibile nella gerarchia
  - Posizionare le operazioni specializzate più in basso
- Descrivere tutti i vincoli applicabili all'operazione
  - Parametri formali, loro tipo e significato
  - Pre-condizioni, post-condizioni, invarianti di classe
  - Eccezioni sollevate
  - Eventi che attivano l'operazione
  - Applicabilità dell'operazione
  - …

Individuazione delle Operazioni

- PRE-CONDIZIONE
  Espressione logica riguardante le aspettative sullo stato del sistema prima che venga eseguita un'operazione
  - Esplicita in modo chiaro che è responsabilità della procedura chiamante controllare la correttezza degli argomenti passati
  - Ad esempio, per l'operazione **`CalcolaRadiceQuadrata(valore)`**,
    la pre-condizione potrebbe essere: "**`valore ≥ 0`**"

Individuazione delle Operazioni

- POST-CONDIZIONE
  Espressione logica riguardante le aspettative sullo stato del sistema dopo l'esecuzione di un'operazione
  - Ad esempio, per l'operazione **`CalcolaRadiceQuadrata(valore)`**,
    la post-condizione potrebbe essere:
    "**`valore == risultato * risultato`**"

Individuazione delle Operazioni

- INVARIANTE di classe
  Vincolo di classe (espressione logica) che deve essere sempre verificato
  - sia all'inizio
  - sia alla fine

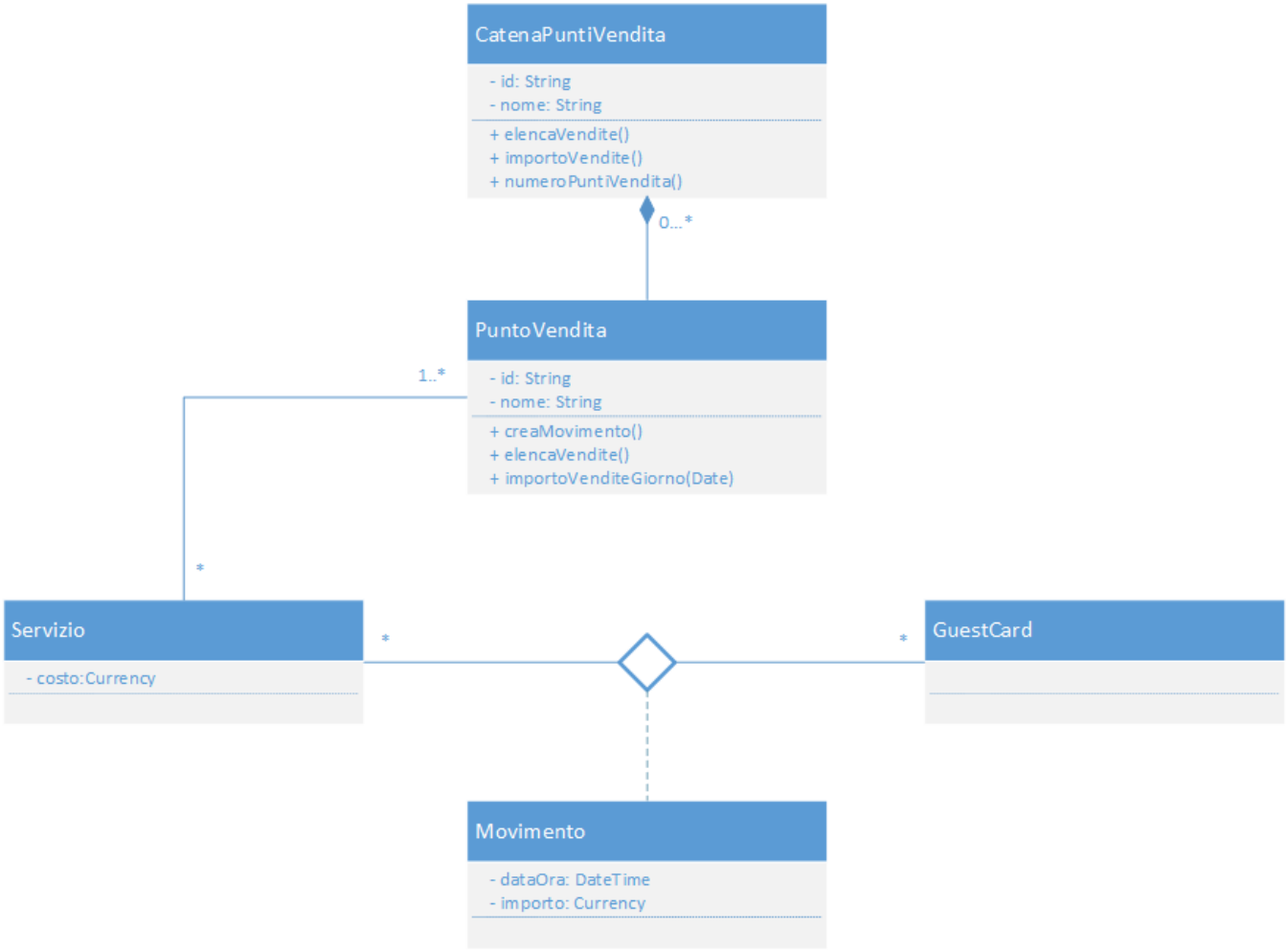
  di tutte le operazioni pubbliche della classe
  Può non essere verificato solo durante l'esecuzione dell'operazione

Individuazione delle Operazioni

- ECCEZIONE
  Si verifica quando un'operazione
  - viene invocata nel rispetto delle sue pre-condizioni
  - ma non è in grado di terminare la propria esecuzione nel rispetto delle post-condizioni

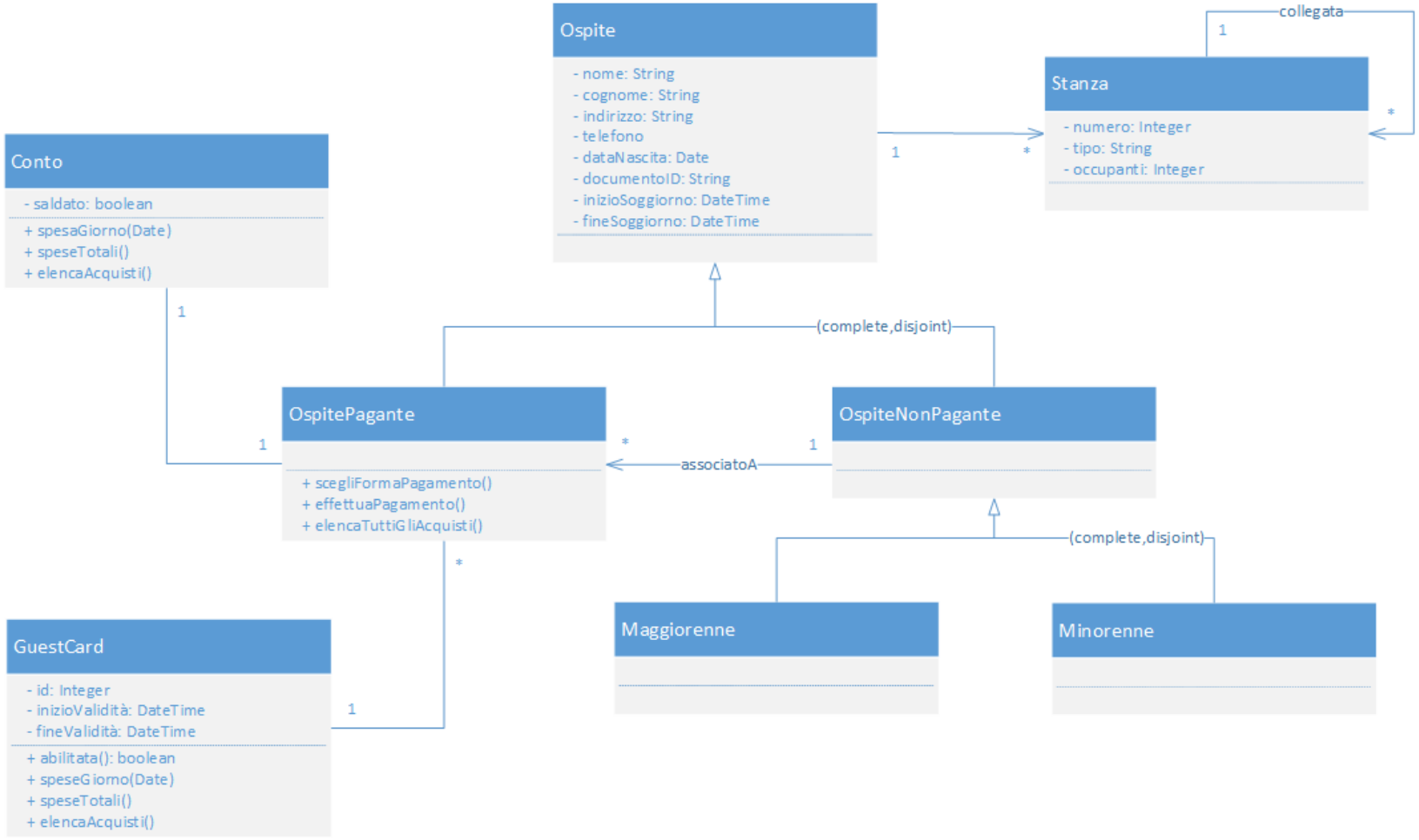
Esempio

- Modello del Dominio “Vendita Servizi” per il Villaggio Turistico



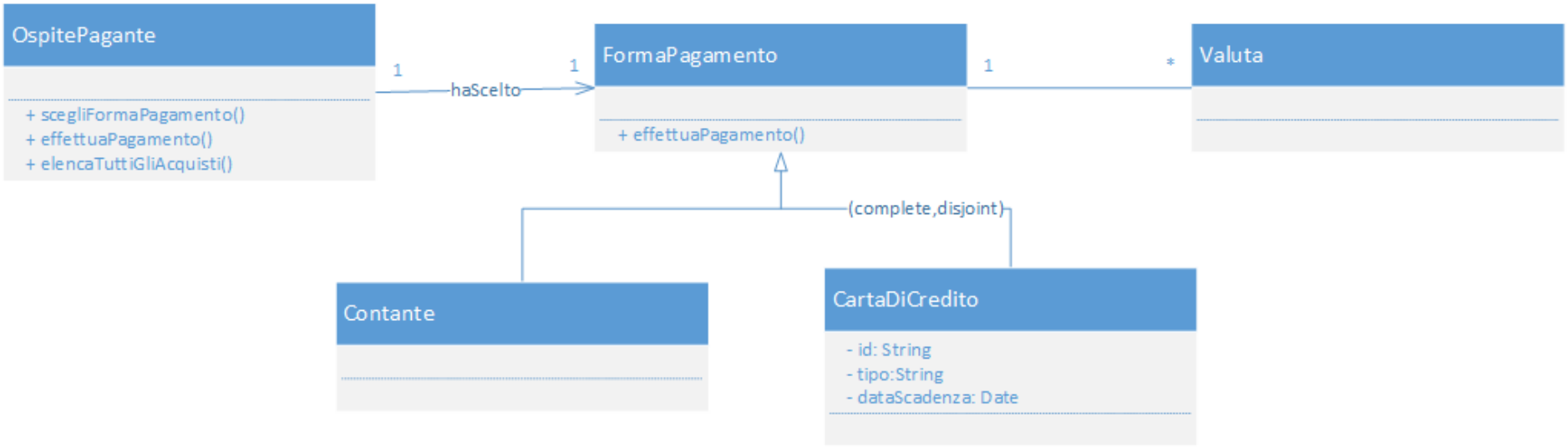
Esempio

- Modello del Dominio “Ospite” per il Villaggio Turistico



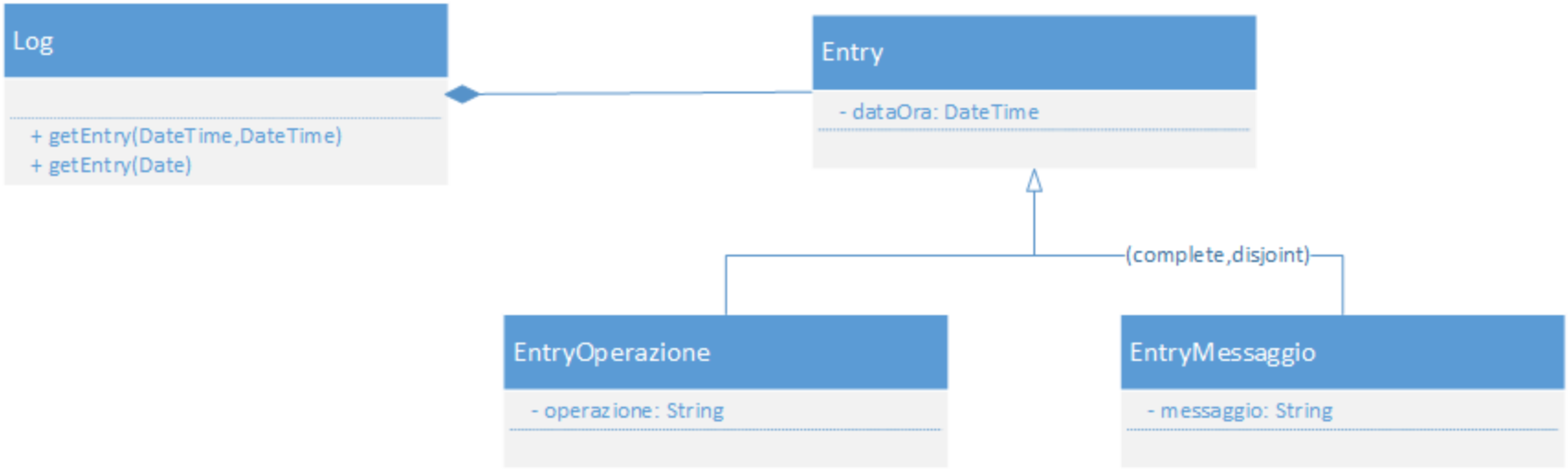
Esempio

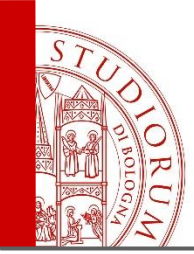
- Modello del Dominio “Pagamenti” per il Villaggio Turistico



Esempio

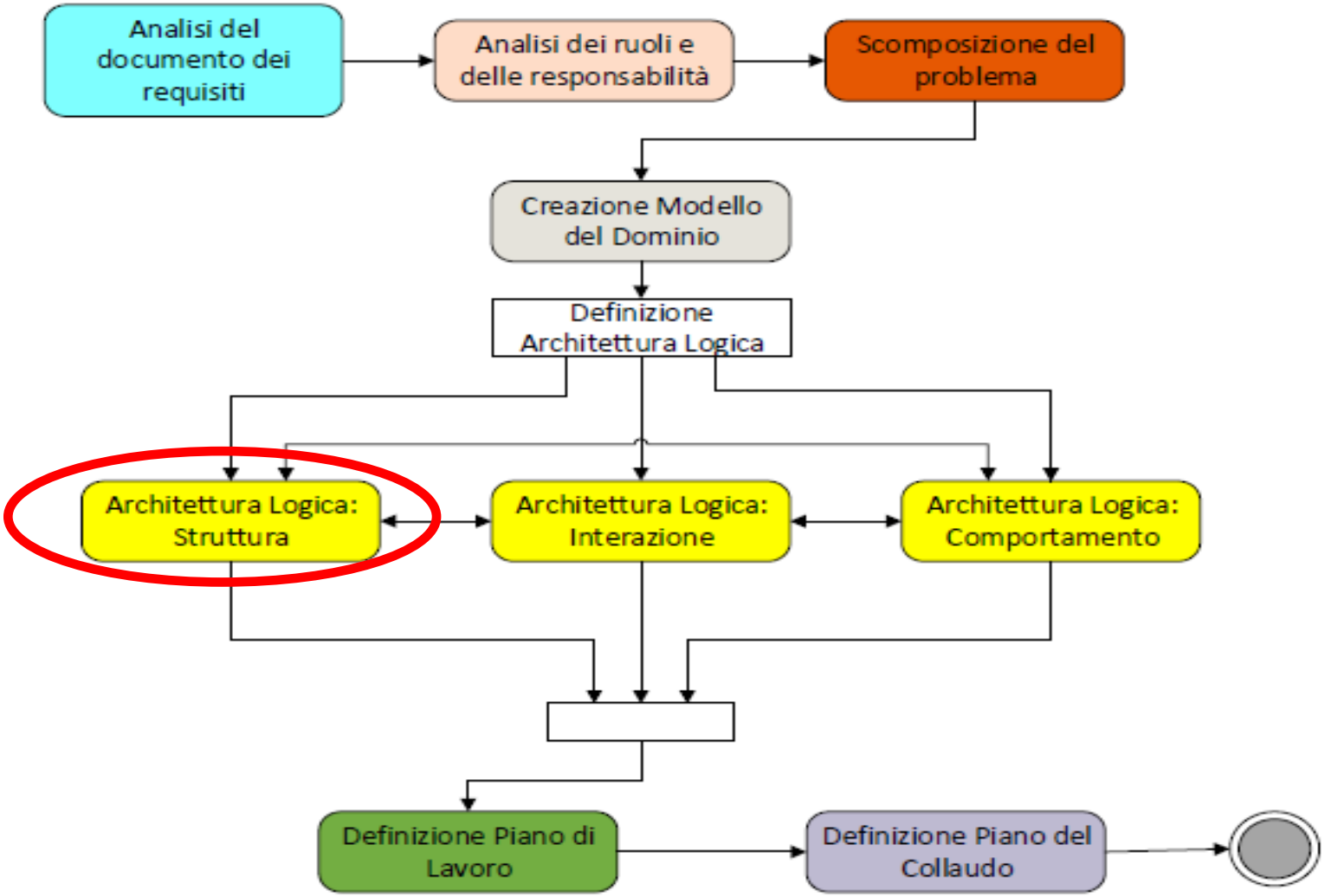
- Modello del Dominio “Log” per il Villaggio Turistico





Architettura Logica: Struttura

Analisi del Problema



Architettura Logica: Struttura

- La parte strutturale dell'Architettura Logica dovrebbe essere composta di due tipi differenti di diagrammi UML
  - *Diagramma dei Package*
    che fornisce una visione di alto livello dell'architettura
  - *Diagramma delle classi* (uno o più diagrammi in base alla complessità) che fornisce una visione più dettagliata del contenuto dei singoli package
- Sarebbe opportuno organizzare sin da subito l'Architettura Logica usando un pattern architetturale chiamato ***Boundary-Control-Entity*** (BCE)

BCE

- BCE è un pattern architetturale che suggerisce di basare l'architettura di un sistema sulla partizione sistematica degli *use case* in oggetti di tre categorie:
  - *informazione*
  - *presentazione*
  - *controllo*

- A ciascuna di queste dimensioni corrisponde uno specifico insieme di classi
- Tale pattern è stato introdotto anche in RUP e sono state adottate icone ben particolari

BCE

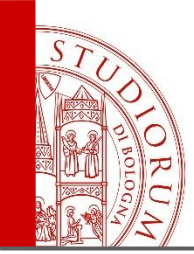
- BCE è un pattern architetturale che suggerisce di basare l'architettura di un sistema sulla partizione sistematica degli *use case* in oggetti di tre categorie:
  - *informazione*
  - *presentazione*
  - *controllo*

- A ciascuna di queste dimensioni corrisponde uno specifico insieme di classi
- Tale pattern è stato introdotto anche in RU e sono state adottate icone ben particolari

Noi useremo una convenzione più semplice → coloriamo package e classi in modo diverso

BCE

- *Entity:* è la dimensione relativa alle entità
  cui corrisponde l'insieme delle classi che includono
  funzionalità relative alle informazioni
  che caratterizzano il problema
  - costituiscono gran parte del modello del dominio
- *Boundary:* è la dimensione relativa alle funzionalità
  che dipendono dall'ambiente esterno cui corrisponde
  l'insieme delle classi che incapsulano l'interfaccia
  del sistema verso il mondo esterno
- *Control: è* la dimensione relativa agli enti
  che incapsulano il controllo
  - il loro compito è di fare da *collante* tra le interfacce e le entità



BCE

- Impostare l'architettura di un sistema software distinguendo tra *boundary*, *control* ed *entity* costituisce un solido punto di partenza per l'organizzazione dell'Architettura Logica di molte applicazioni
- L'analista tende ad affrontare la complessità dei problemi partizionando i problemi stessi in sotto-problemi
- È del tutto logico che un analista eviti di associare alle entità di un dominio
  - sia le funzionalità di una specifica applicazione
  - sia le funzionalità tipiche della interazione con l'utente

BCE

- La conseguenza è che l'architettura
  di un sistema software risulta quasi fisiologicamente
  articolata in una sequenza di livelli (*layer*) verticali
  che viene tipicamente mantenuta
  anche in fase di progetto e implementazione

Layer

- Naturale separare la parte che realizza
  le entità dell'applicazione (*dominio*)
  dalla parte che realizza l'interazione con l'utente (*logica di presentazione*), introducendo una parte centrale
  di connessione (*control*)

Layer

- Nello specifico:
  - il livello di ***presentazione***
    comprende le parti che realizzano l'interfaccia utente
    - Per aumentare la riusabilità delle parti, questo livello è progettato e costruito astraendo quanto più possibile dai dettagli degli specifici dispositivi di I/O
  - il livello di ***applicazione***
    comprende le parti che provvedono a elaborare l'informazione di ingresso, a produrre i risultati attesi e a presentare le informazioni in uscita
  - il livello delle ***entità***
    forma il (modello del) dominio applicativo

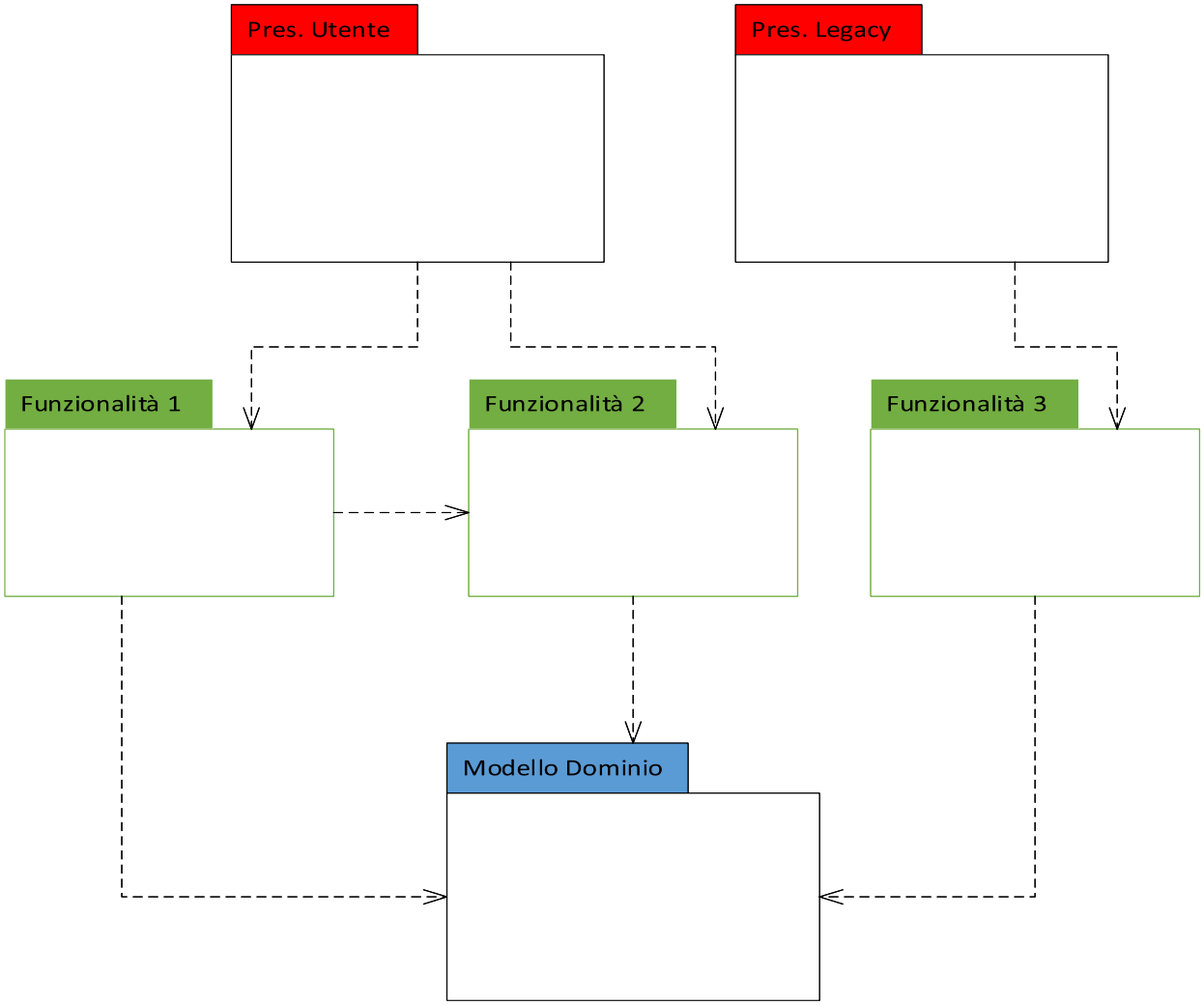
Struttura: Package

- Usando come base di partenza il lavoro di analisi svolto nelle fasi precedenti
  e tenendo in considerazione il pattern BCE
  si può iniziare la creazione del Diagramma dei Package che rappresenta la visione dal alto livello dell'Architettura Logica

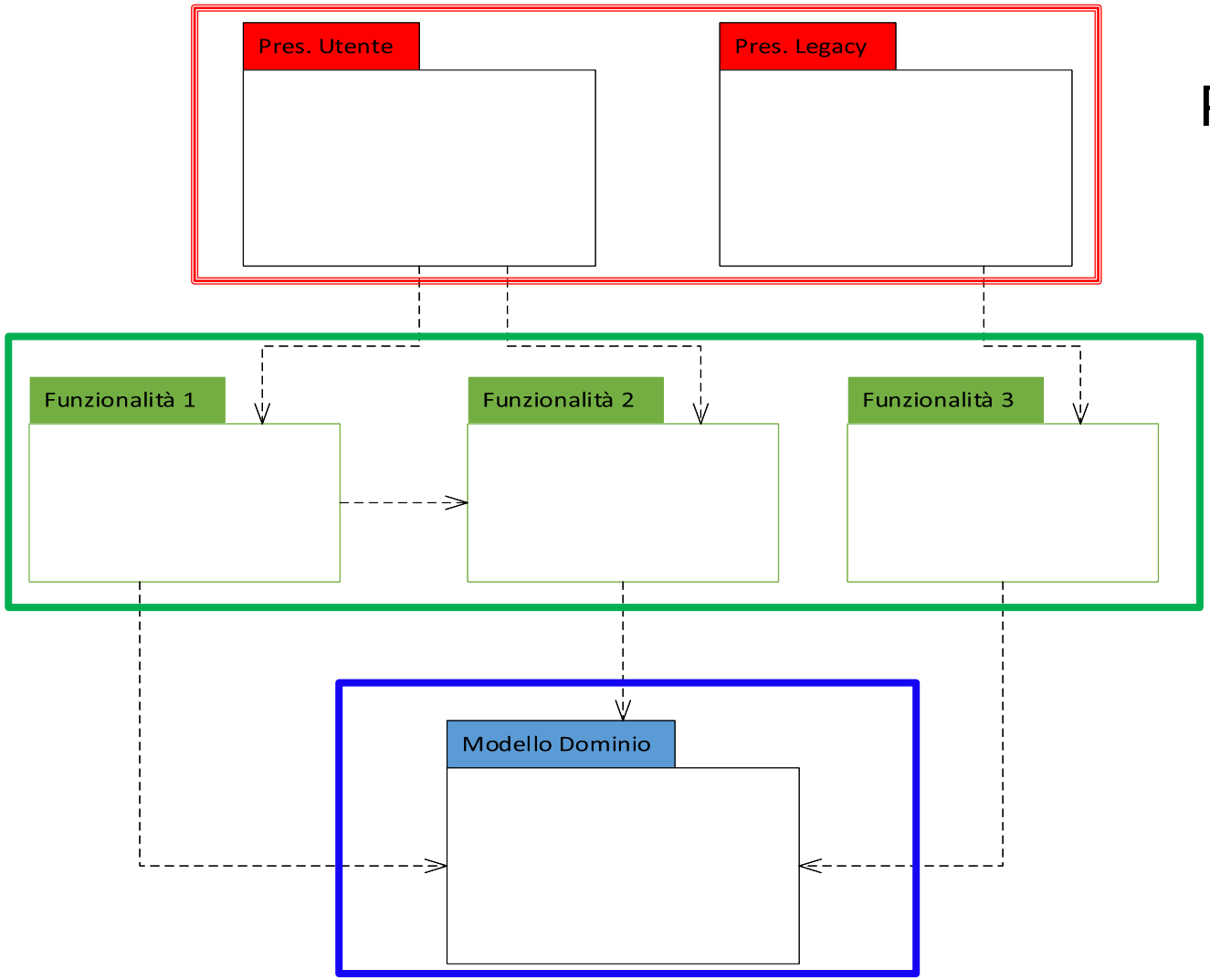
Struttura: Package

- Il primo package che si può identificare
  è il package costituito dal Modello del Dominio
  creato nella fase precedente
  - tale Modello (se ben realizzato) costituisce la parte "entity" dell'architettura
- Poi è possibile creare un package
  per ognuna delle diverse funzionalità identificate
  nella Tabella delle Funzionalità ("control")
- Vengono creati uno o più package
  per la parte di "boundary"
- Infine si identificano le *dipendenze logiche* tra i package

Struttura: Package



Struttura: Package

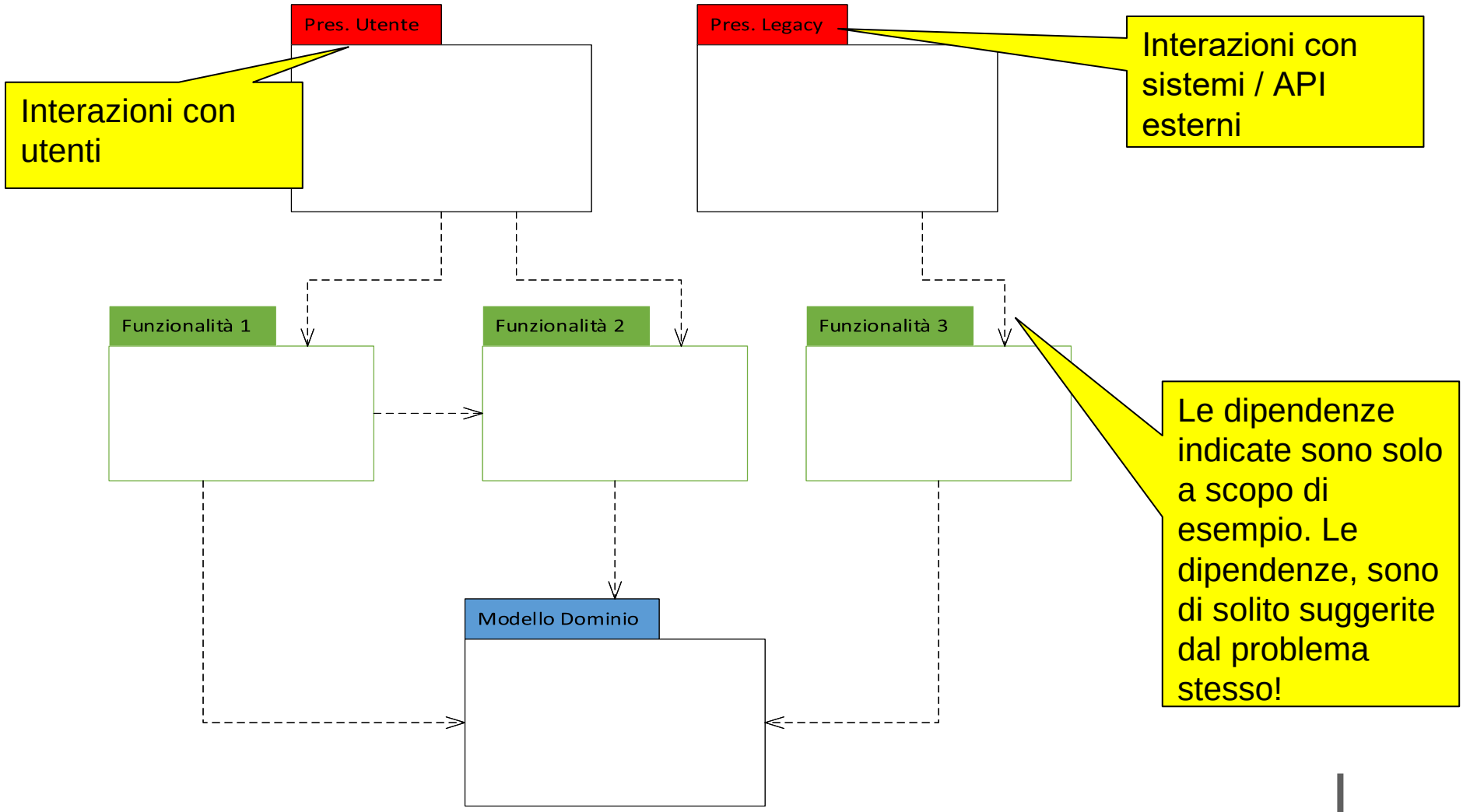


Presentazione

Applicazione

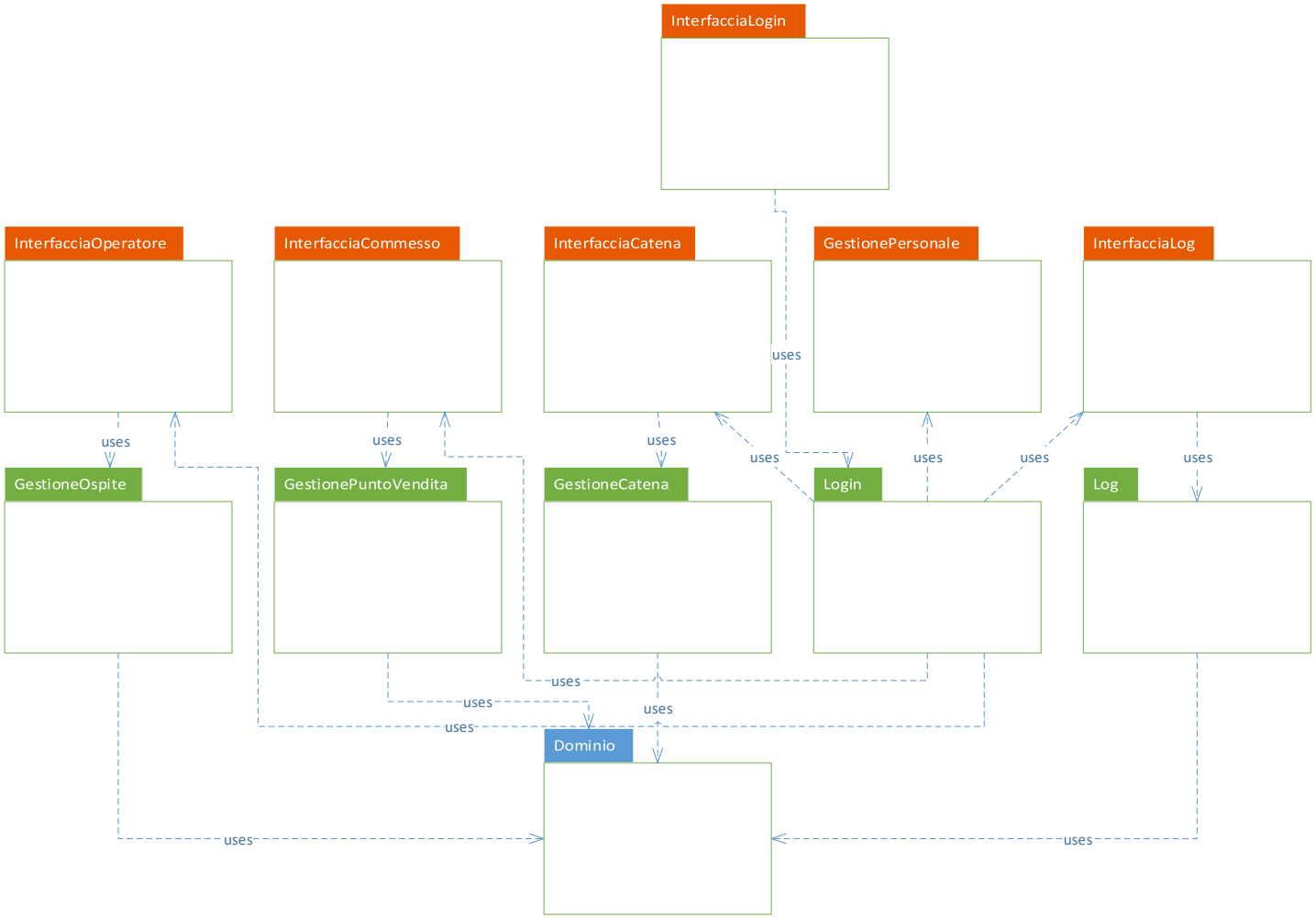
Entità

Struttura: Package



Esempio

- Diagramma dei Package per il Villaggio Turistico



Struttura: Classi

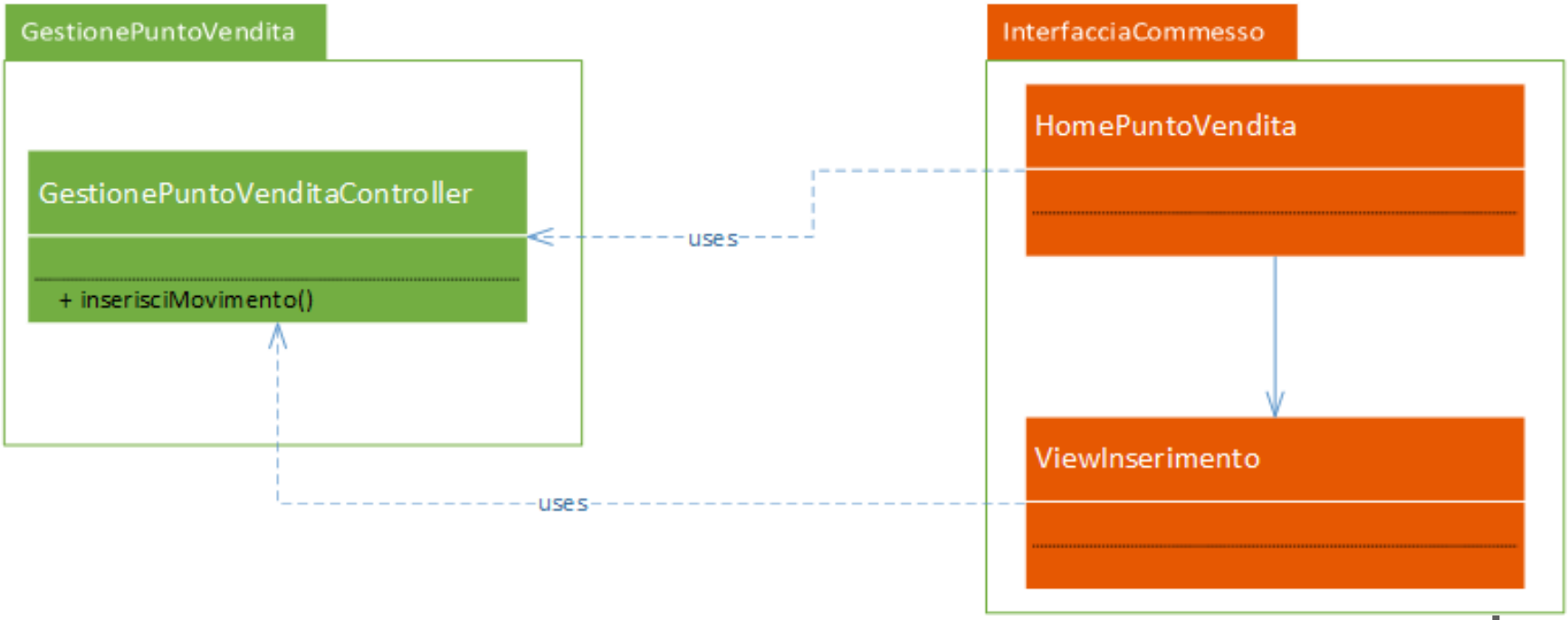
- Dopo aver stabilito la struttura di alto livello dell'Architettura Logica
  si dettagliano le classi che compongono ogni package
- Il Package del Dominio è già stato identificato nel Modello del Dominio
- Se le classi da specificare per ogni package sono poche si può realizzare un unico diagramma delle classi
- Altrimenti è possibile creare un diagramma delle classi separato per ogni package

Struttura: Classi

- Attenzione a *non introdurre scelte di progettazione* in questa fase
- Indicare solo quelle classi che sono deducibili dal problema
  - in genere viene inserita almeno una classe che realizza le funzionalità indicate nelle specifiche
  - se nel diagramma dei package è stata indicata una funzionalità che nella fase precedente era stata poi scomposta si possono indicare le classi che realizzano le sotto-funzionalità
  - indicare le classi che rappresentano le maschere di interazione con l'utente identificate nelle fasi precedenti

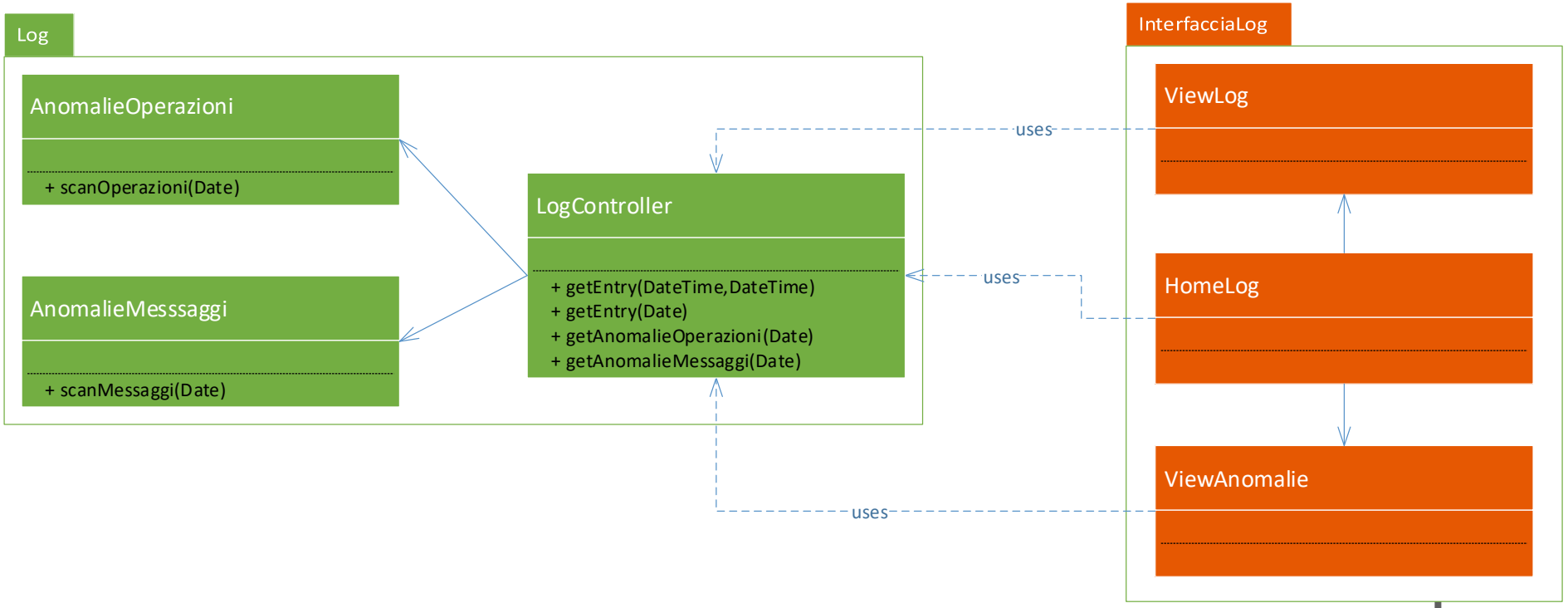
Esempio

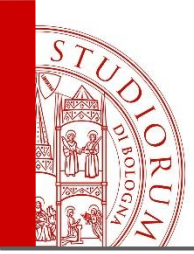
- Diagramma delle classi per il Villaggio Turistico
  - Diagramma delle classi: InterfacciaCommesso & GestionePuntoVendita



Esempio

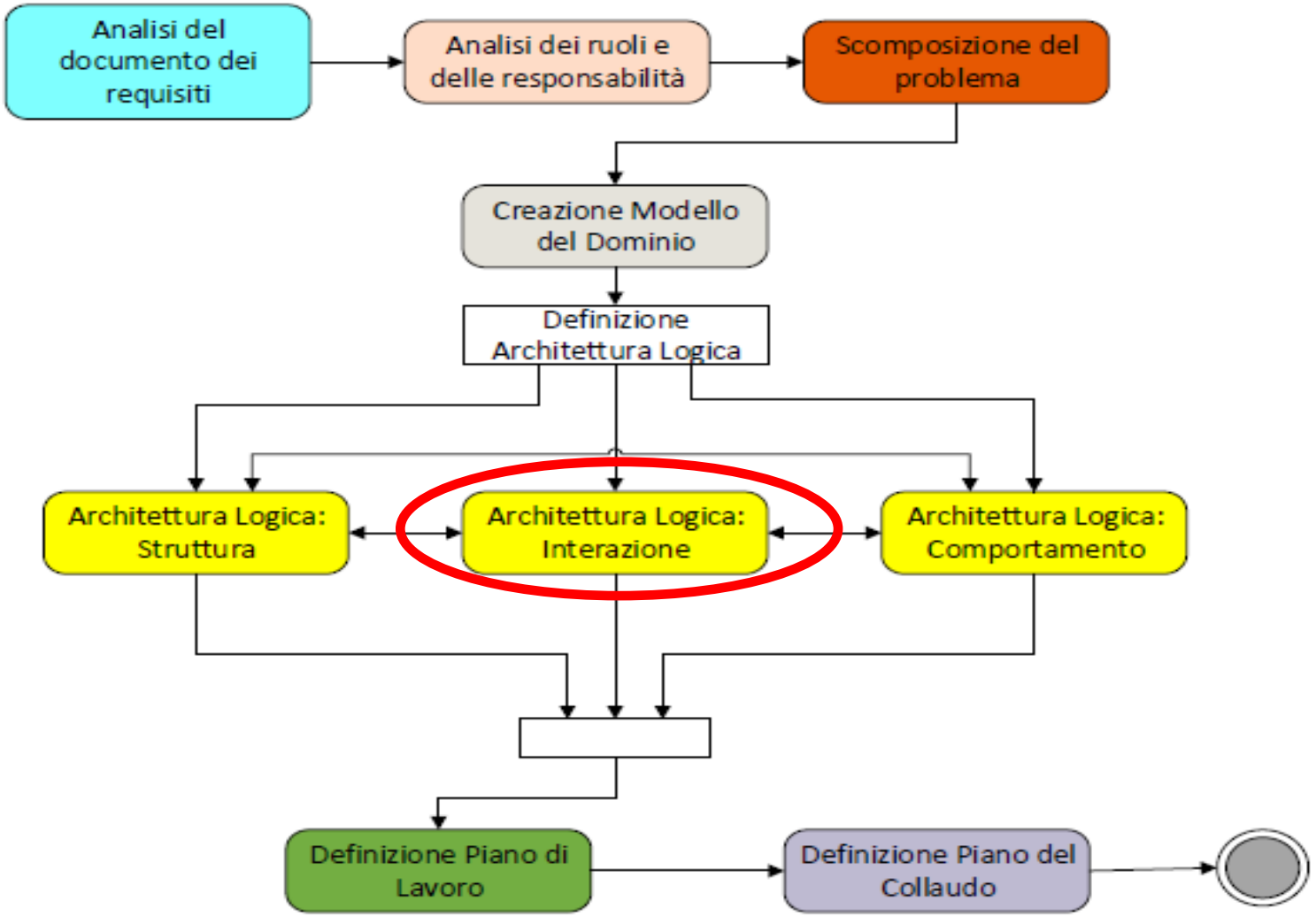
- Diagramma delle classi per il Villaggio Turistico
  - Diagramma delle classi: InterfacciaLog & Log





Architettura Logica: Interazione

Analisi del Problema



Architettura Logica: Interazione

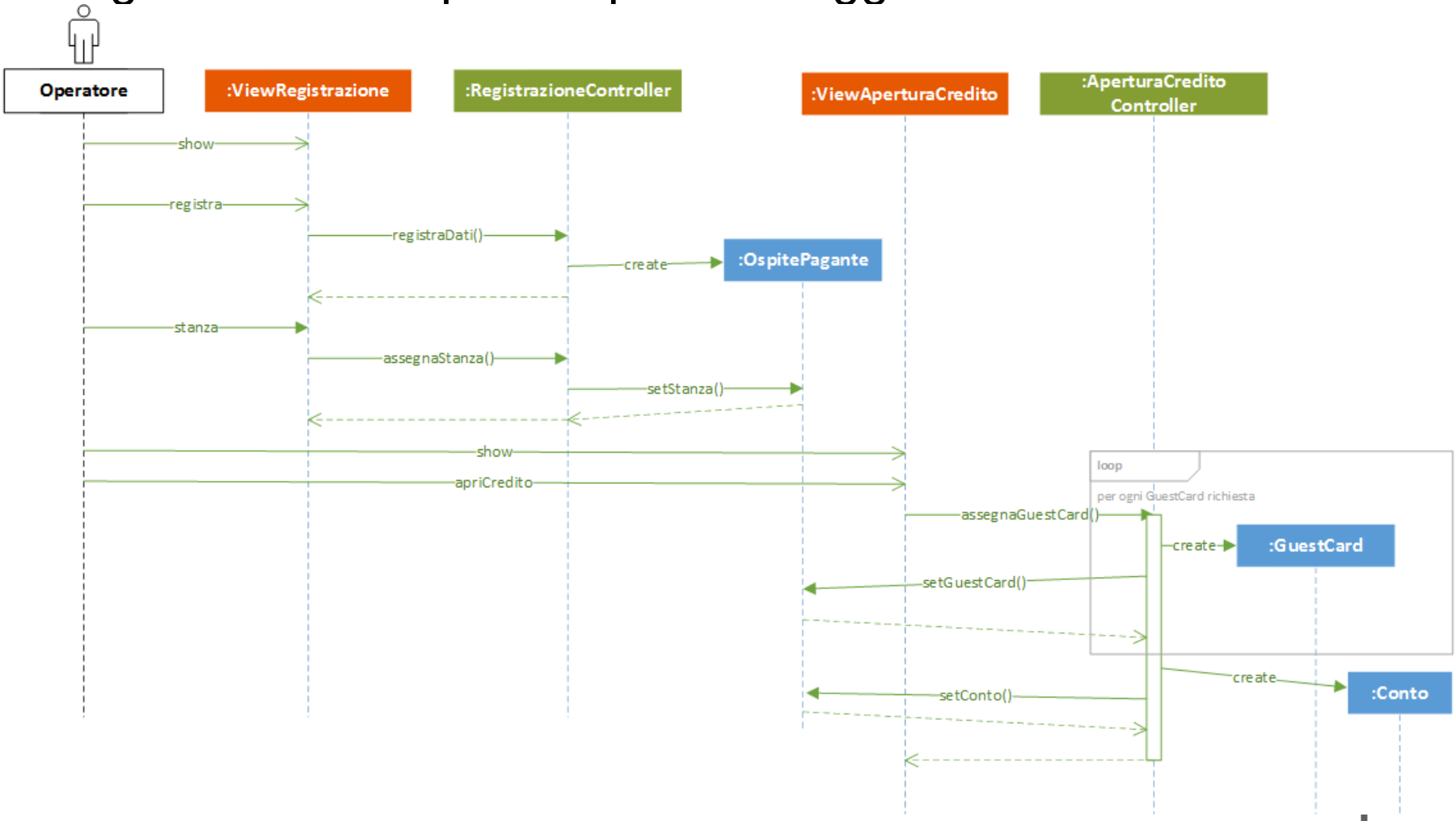
- Descrivere le interazioni tra le entità identificate nella parte strutturale attraverso opportuni Diagrammi di Sequenza
- Diagrammi di sequenza - evidenziano
  - lo scambio di messaggi (le interazioni) tra gli oggetti
  - l'ordine in cui i messaggi vengono scambiati tra gli oggetti (sequenza di invocazioni delle operazioni)

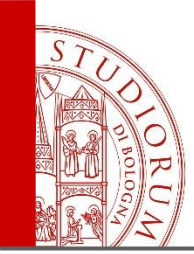
Architettura Logica: Interazione

- Non spingere la definizione dei diagrammi di sequenza sino ai minimi dettagli
- Utilizzare i diagrammi solo per descrivere il funzionamento del sistema
  - in risposta a sollecitazioni esterne
  - in fasi particolarmente significative
  - nei casi più critici

Esempio

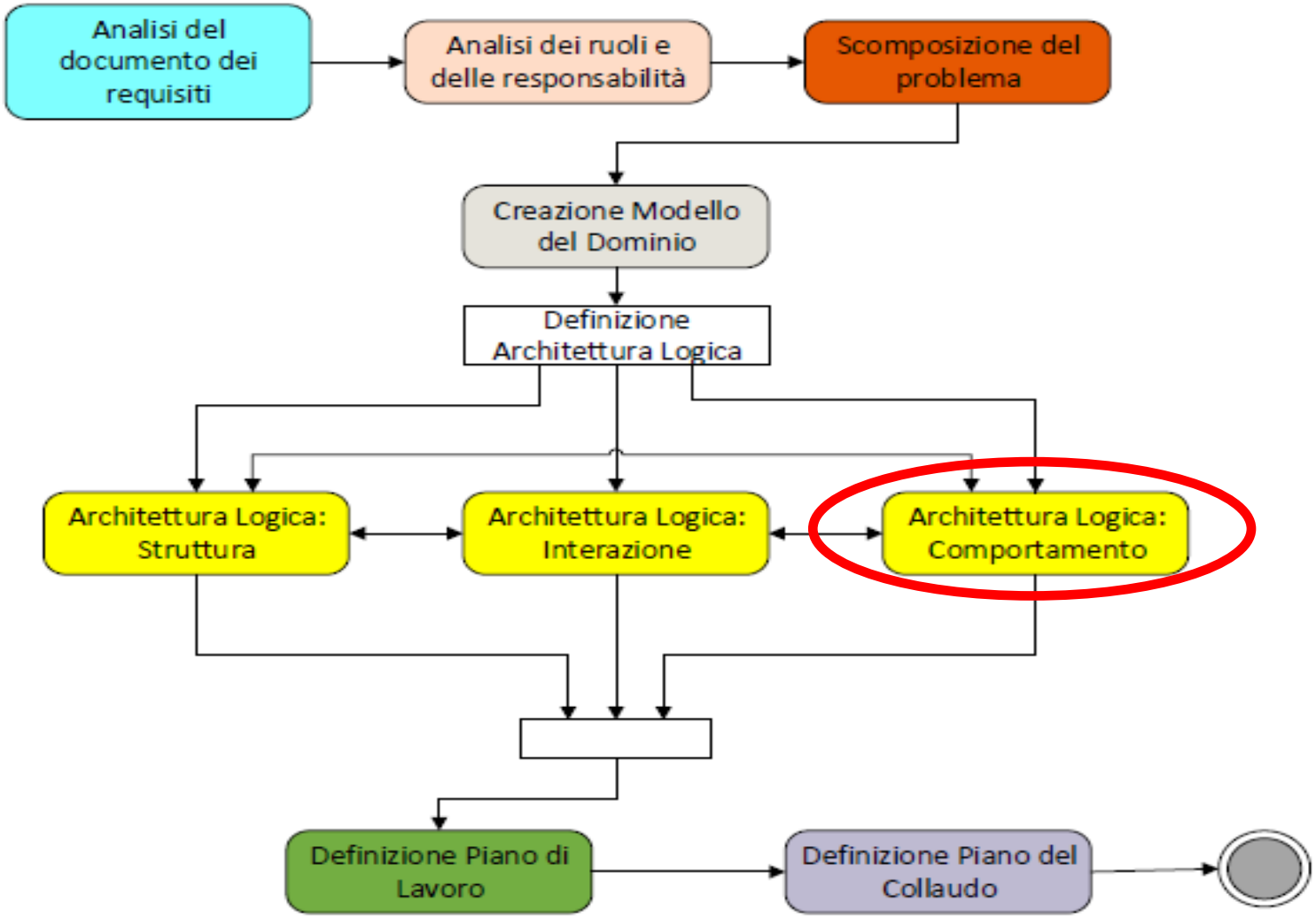
- Diagramma di sequenza per il Villaggio Turistico





Architettura Logica: Comportamento

Analisi del Problema



Architettura Logica: Comportamento

- Descrivere il comportamento delle entità identificate nella parte strutturale attraverso opportuni Diagrammi di Stato o delle Attività
- Diagramma di Stato per mostrare come si comportano alcune entità complesse a seguito delle interazioni e degli eventi che avvengono nel sistema
- Diagramma delle Attività per dettagliare funzionamenti complessi delle entità
- Non spingere la definizione dei diagrammi sino ai minimi dettagli

Architettura Logica: Comportamento

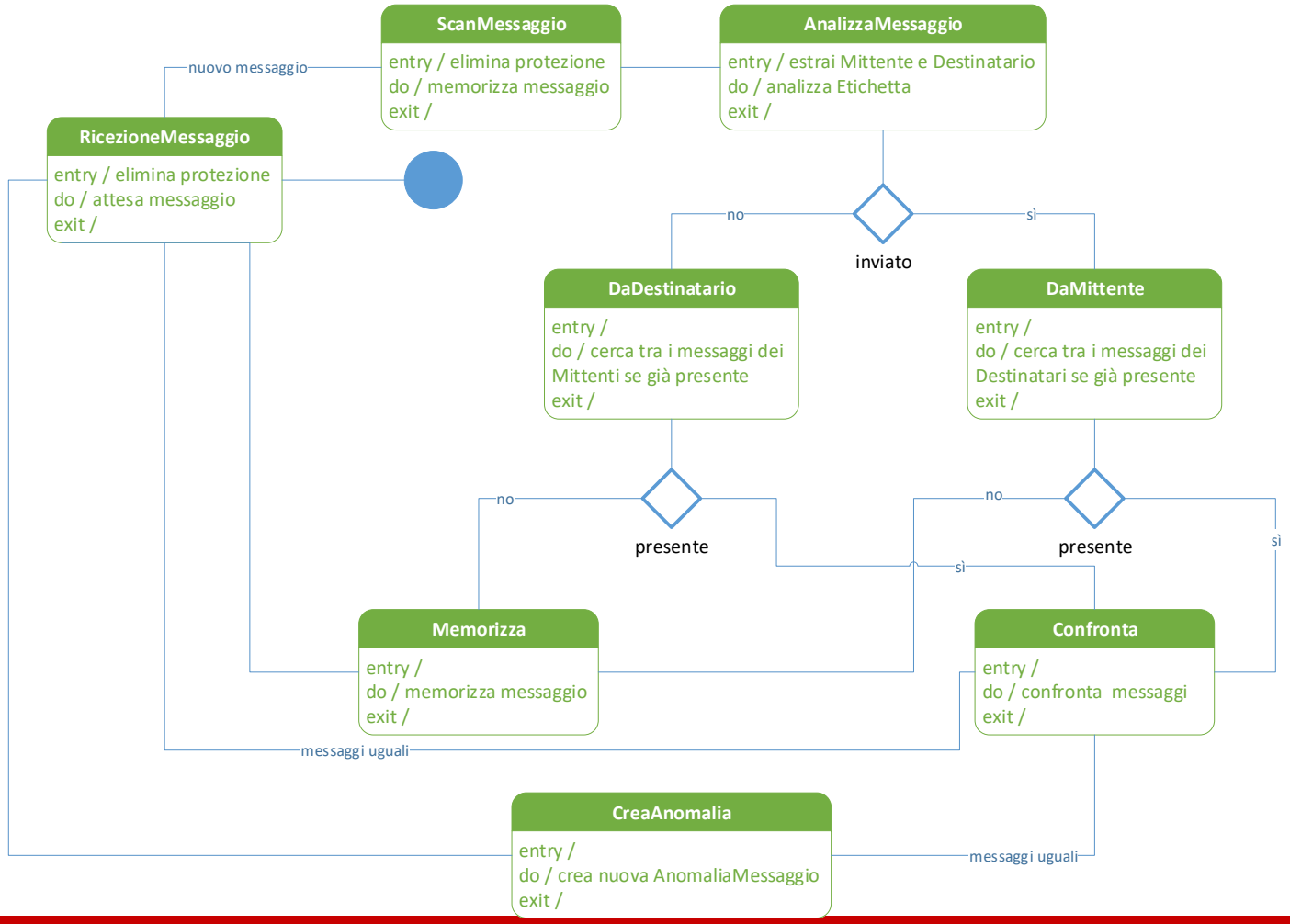
- Lo stato di un oggetto è dato dal valore dei suoi attributi e delle sue associazioni
- In molti domini applicativi, esistono oggetti che, a seconda del proprio stato, rispondono in maniera diversa ai messaggi ricevuti
  - Dispositivi (spento, in attesa, operativo, guasto, ecc.)
  - Transazioni complesse (in definizione, in esecuzione, completata, fallita, ecc.)
- In questi casi, è opportuno disegnare un diagramma di stato per l'oggetto, mostrando i possibili stati e gli eventi che attivano transizioni da uno stato all'altro

Architettura Logica: Comportamento

- A un oggetto possono essere assegnate responsabilità che comportano un insieme di elaborazioni complesse che devono essere eseguite in un ordine particolare
- In questi casi, è opportuno disegnare un diagramma di attività per l'oggetto, mostrando le diverse elaborazioni che devono essere portate a termine e l'ordine di tali elaborazioni

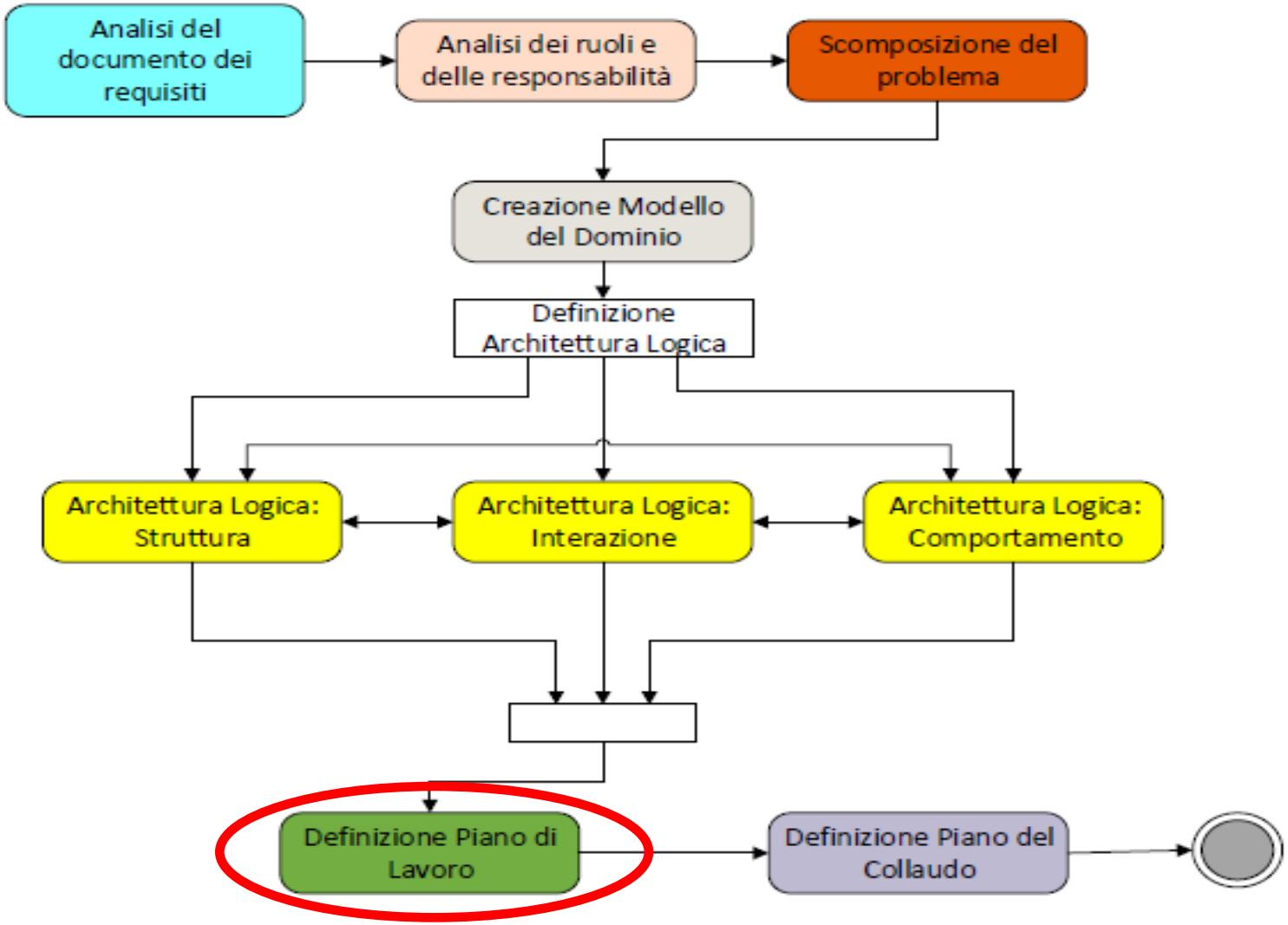
Esempio

- Diagramma di stato per AnomalieMessaggi



Definizione del Piano di Lavoro

Analisi del Problema



Definizione Piano di Lavoro

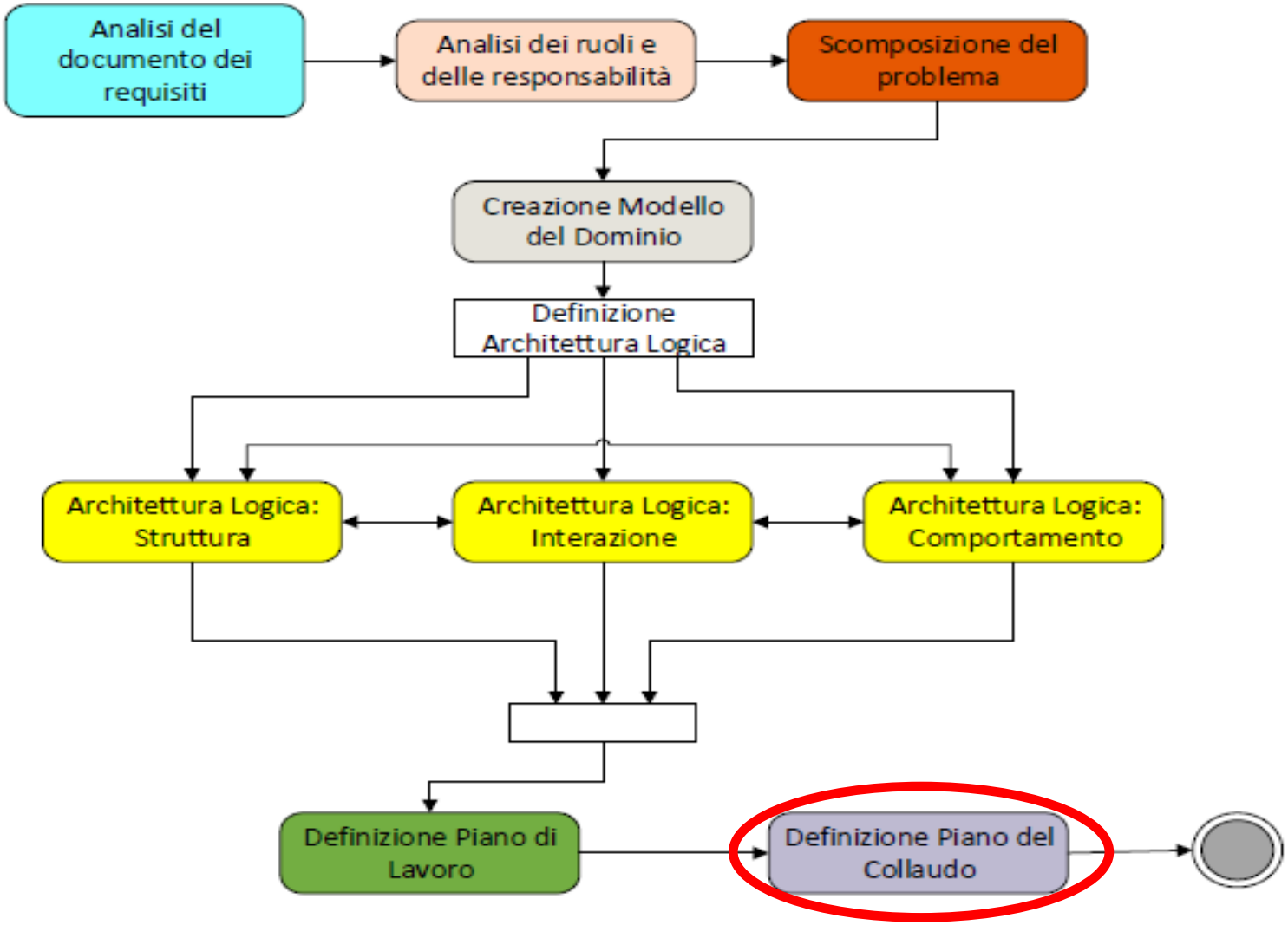
- Dopo la creazione dell'Architettura Logica è possibile iniziare a suddividere il lavoro
- In particolare è necessario:
  - suddividere le responsabilità ai diversi membri del team di progetto e sviluppo
  - stabilire le tempistiche per la progettazione di ciascuna parte
  - stabilire i tempi di sviluppo di ciascun sotto-sistema
  - programmare i test di integrazione tra le parti
  - identificare i tempi di rilascio delle diverse versioni del prototipo
  - identificare un piano per gli sviluppi futuri

Esempio

Package	Progetto	Sviluppo
Dominio	Team progettazione + Team DB	Team sviluppo A + Team DB
Gestione Ospite	Team progettazione	Team sviluppo A
GestionePuntoVendita	Team progettazione	Team sviluppo B
GestioneCatena	Team progettazione	Team sviluppo B
Login	Team sicurezza	Team sviluppo sicurezza
Log	Team sicurezza	Team sviluppo sicurezza
InterfacciaOperatore	Team progettazione + Team grafico	Team sviluppo A + Team Grafico
InterfacciaCommesso	Team progettazione + Team grafico	Team sviluppo B + Team Grafico
InterfacciaCatena	Team progettazione +Team grafico	Team sviluppo B + Team Grafico
InterfacciaLogin	Team sicurezza+ Team grafico	Team sicurezza+ Team grafico
InterfacciaLog	Team sicurezza+ Team grafico	Team sicurezza+ Team grafico
GestionePersonale	Team progettazione + Team DB	Team sviluppo A + Team DB

Definizione del Piano del Collaudo

Analisi del Problema



Definizione Piano del Collaudo

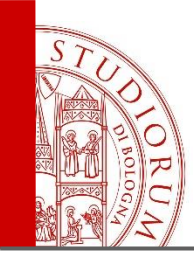
- Al termine dell'Analisi del Problema, i modelli che definiscono il dominio e l'Architettura Logica dovrebbero dare sufficienti informazioni su ***cosa*** le varie parti del sistema debbano fare senza specificare ancora molti dettagli del loro comportamento
- Il "***cosa fare***" di una parte dovrà comprendere anche le forme di interazione con le altre parti
- Lo scopo del ***piano del collaudo*** è cercare di precisare il comportamento atteso da parte di una entità prima ancora di iniziarne il progetto e la realizzazione
- Focalizzando l'attenzione sulle interfacce delle entità e sulle interazioni è possibile impostare scenari in cui specificare in modo già piuttosto dettagliato la "risposta" di una parte a uno "stimolo" di un'altra parte

Definizione Piano del Collaudo

- Lo sforzo di definire nel modo più preciso possibile un piano del collaudo di un sistema prima ancora di averne iniziato la fase di progettazione viene ricompensato da
  - una miglior comprensione dei requisiti
  - un approfondimento nella comprensione dei problemi
  - una più precisa definizione dell'insieme delle funzionalità (operazioni) che ciascuna parte deve fornire alle altre per una effettiva integrazione nel "tutto" che costituirà il sistema da costruire
  - comprendere il significato delle entità e specificarne nel modo più chiaro possibile il comportamento atteso

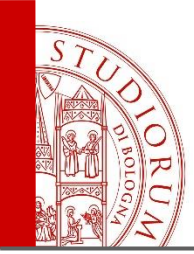
Definizione Piano del Collaudo

- Un piano del collaudo va concepito e impostato da un punto di ***vista logico***, cercando di individuare categorie di comportamenti e punti critici
- In molti casi tuttavia può anche risultare possibile definire in modo precoce *piani di collaudo concretamente eseguibili*, avvalendosi di strumenti del tipo JUnit/NUnit che sono ormai diffusi in tutti gli ambienti di programmazione
- Lo sforzo di definire un piano di collaudo concretamente eseguibile promuove uno sviluppo controllato, sicuro e consapevole del codice poiché il progettista e lo sviluppatore possono verificare subito in modo concreto la correttezza di quanto sviluppato



JUnit

- JUnit (https://junit.org/junit5/docs/current/user-guide/) è un framework per unit-testing per il linguaggio Java
- Dovreste conoscerlo già bene da Fondamenti T-2



NUnit

- NUnit (http://nunit.org/) è un framework per unit-testing per tutti i linguaggi .Net
- NUnit è inizialmente derivato da JUnit,
  ma è stato totalmente riscritto dalla versione 3
- Troverete la documentazione e la guida all'installazione a
  https://github.com/nunit/docs/wiki/Framework-Release-Notes